

Milena (Olena)
User documentation 1.0a Id

Generated by Doxygen 1.7.1

Thu Sep 15 2011 15:47:21

Contents

1	Documentation of milena	1
1.1	Introduction	1
1.2	Overview of Milena.	1
1.3	Copyright and License.	2
2	Quick Reference Guide	3
2.1	Installation	4
2.1.1	Requirements	4
2.1.1.1	To compile the user examples	4
2.1.1.2	To compile the documentation (Optional)	4
2.1.1.3	To develop in Olena	4
2.1.2	Getting Olena	4
2.1.3	Building Olena	4
2.2	Foreword	4
2.2.1	Generality	4
2.2.2	Directory hierarchy	4
2.2.3	Writing and compiling a program with Olena	4
2.3	Site	4
2.4	Site set	4
2.4.1	Basic interface	4
2.4.2	Optional interface	4
2.5	Image	5
2.5.1	Definition	5
2.5.2	Possible image types	5
2.5.3	Possible value types	5
2.5.4	Domain	5
2.5.5	Border and extension	6
2.5.5.1	Image border	6

2.5.5.2	Generality on image extension	6
2.5.5.3	Different extensions	6
2.5.6	Interface	8
2.5.7	Load and save images	8
2.5.8	Create an image	8
2.5.9	Access and modify values	9
2.5.10	Image size	9
2.6	Structural elements: Window and neighborhood	10
2.6.1	Define an element	10
2.6.1.1	Window	10
2.6.1.2	Neighborhood	10
2.6.1.3	Custom structural elements	10
2.6.1.4	Conversion between Neighborhoods and Windows	10
2.7	Sites, psites and dpoints	10
2.7.1	Need for site	10
2.7.2	Need for psite	10
2.7.3	From psite to site	11
2.7.4	Dpoint	11
2.8	Iterators	11
2.9	Memory management	12
2.10	Basic routines	12
2.10.1	Fill	12
2.10.2	Paste	12
2.10.3	Blobs	13
2.10.4	Logical not	13
2.10.5	Compute	13
2.10.5.1	Accumulators	13
2.10.5.2	Example with labeling::compute()	13
2.10.6	Working with parts of an image	14
2.10.6.1	Restrict an image with a site set	14
2.10.6.2	Restrict an image with a predicate	15
2.10.6.3	Restrict an image with a C function	15
2.11	Input / Output	16
2.11.1	ImageMagick	16
2.11.2	GDCM	16
2.12	Graphs and images	16

2.12.1	Description	16
2.12.2	Example	16
2.13	Useful global variables	19
2.14	Useful macros	19
2.14.1	Variable declaration macros	19
2.14.2	Iterator type macros	19
2.14.2.1	Default iterator types	19
2.14.2.2	Forward iterator types	19
2.14.2.3	Backward iterators	19
2.14.2.4	Graph iterators	19
2.15	Common Compilation Errors	19
3	Tutorial	21
3.1	Welcome	21
3.1.1	How to learn Milena	21
3.1.2	Obtaining the library	21
3.1.3	Downloading the library	21
3.1.3.1	Downloading from SVN	21
3.1.3.2	Downloading packaged releases	21
3.1.4	Join the mailing lists	22
3.1.5	Directory structure	22
3.1.6	Documentation	22
3.1.7	Community and Support	22
3.1.8	Project status	22
3.1.9	A brief history of Milena	22
3.1.10	Contacts	22
3.2	Installation	22
3.2.1	Bootstrap (SVN Sources)	22
3.2.2	Configure	22
3.2.3	Install	22
3.2.4	Optional compilation	22
3.2.4.1	Examples	22
3.2.4.2	Tools	22
3.2.4.3	Tests	22
3.2.5	Installation content	22
3.3	Getting started with Milena	23
3.3.1	Getting familiar with genericity	23

3.3.2	First generic algorithm	23
3.3.3	Compilation	25
3.3.3.1	Include path	25
3.3.3.2	Library linking	25
3.3.3.3	Disable Debug	25
3.3.3.4	Compiler optimization flags	25
3.3.4	Debug hints	25
3.3.4.1	Using assertions and GDB	25
3.3.4.2	Traces	25
3.3.4.3	Debug routines	25
3.4	Data representation	26
3.4.1	Sites	26
3.4.2	Site sets	26
3.4.2.1	Creating a site set	26
3.4.2.2	Getting access to sites	27
3.4.3	Images	27
3.4.3.1	Creating an image	27
3.4.3.2	Reading an image from a file	27
3.4.3.3	Accessing data	27
3.5	Load and save images	27
3.6	Create your first image	27
3.7	Read and write images	28
3.8	Regions of interest	28
3.8.1	Image domain restricted by a site set	29
3.8.2	Image domain restricted by a function	29
3.8.3	Image domain restricted by a mask	29
3.8.4	Image domain restricted by a predicate	29
4	Module Index	31
4.1	Modules	31
5	Namespace Index	33
5.1	Namespace List	33
6	Class Index	37
6.1	Class Hierarchy	37
7	Class Index	81

7.1	Class List	81
8	Module Documentation	91
8.1	On site sets	91
8.1.1	Detailed Description	91
8.2	On images	91
8.2.1	Detailed Description	92
8.3	On values	92
8.3.1	Detailed Description	94
8.4	Multiple accumulators	94
8.4.1	Detailed Description	94
8.5	Graphes	94
8.5.1	Detailed Description	95
8.6	Images	95
8.6.1	Detailed Description	95
8.7	Basic types	95
8.7.1	Detailed Description	96
8.8	Image morphers	96
8.9	Values morphers	96
8.9.1	Detailed Description	97
8.10	Domain morphers	97
8.10.1	Detailed Description	97
8.11	Identity morphers	97
8.11.1	Detailed Description	98
8.12	Types	98
8.12.1	Detailed Description	99
8.13	Accumulators	99
8.13.1	Detailed Description	99
8.14	Routines	99
8.15	Canvas	99
8.16	Functions	99
8.16.1	Detailed Description	101
8.17	Neighborhoods	101
8.17.1	Detailed Description	101
8.18	1D neighborhoods	101
8.18.1	Detailed Description	101
8.18.2	Typedef Documentation	102

8.18.2.1	neighb1d	102
8.18.3	Function Documentation	102
8.18.3.1	c2	102
8.19	2D neighborhoods	102
8.19.1	Detailed Description	103
8.19.2	Typedef Documentation	103
8.19.2.1	neighb2d	103
8.19.3	Function Documentation	103
8.19.3.1	c2_col	103
8.19.3.2	c2_row	103
8.19.3.3	c4	103
8.19.3.4	c8	104
8.20	3D neighborhoods	104
8.20.1	Detailed Description	105
8.20.2	Typedef Documentation	105
8.20.2.1	neighb3d	105
8.20.3	Function Documentation	105
8.20.3.1	c18	105
8.20.3.2	c26	105
8.20.3.3	c2_3d_sli	106
8.20.3.4	c4_3d	106
8.20.3.5	c6	107
8.20.3.6	c8_3d	107
8.21	Site sets	108
8.21.1	Detailed Description	108
8.22	Basic types	108
8.22.1	Detailed Description	108
8.23	Graph based	109
8.23.1	Detailed Description	109
8.24	Complex based	109
8.24.1	Detailed Description	109
8.25	Sparse types	109
8.25.1	Detailed Description	110
8.26	Queue based	110
8.26.1	Detailed Description	110
8.27	Utilities	110

8.27.1 Detailed Description	111
8.28 Windows	111
8.28.1 Detailed Description	112
8.29 1D windows	112
8.29.1 Detailed Description	112
8.29.2 Typedef Documentation	112
8.29.2.1 segment1d	112
8.29.2.2 window1d	113
8.30 2D windows	113
8.30.1 Detailed Description	114
8.30.2 Typedef Documentation	114
8.30.2.1 disk2d	114
8.30.2.2 hline2d	114
8.30.2.3 vline2d	114
8.30.2.4 window2d	115
8.30.3 Function Documentation	115
8.30.3.1 win_c4p	115
8.30.3.2 win_c8p	115
8.31 3D windows	115
8.31.1 Detailed Description	116
8.31.2 Typedef Documentation	116
8.31.2.1 sline3d	116
8.31.2.2 sphere3d	117
8.31.2.3 window3d	117
8.31.3 Function Documentation	117
8.31.3.1 win_c4p_3d	117
8.31.3.2 win_c8p_3d	117
8.32 N-D windows	118
8.32.1 Detailed Description	118
8.33 Multiple windows	118
8.33.1 Detailed Description	118
8.34 v2w2v functions	118
8.35 v2w_w2v functions	119
8.36 vv2b functions	119
9 Namespace Documentation	121
9.1 mln Namespace Reference	121

9.1.1	Detailed Description	142
9.1.2	Typedef Documentation	144
9.1.2.1	bin_1complex_image2d	144
9.1.2.2	bin_2complex_image3df	144
9.1.2.3	box1d	144
9.1.2.4	box2d	144
9.1.2.5	box2d_h	145
9.1.2.6	box3d	145
9.1.2.7	discrete_plane_1complex_geometry	145
9.1.2.8	discrete_plane_2complex_geometry	145
9.1.2.9	dpoint1d	145
9.1.2.10	dpoint2d	145
9.1.2.11	dpoint2d_h	146
9.1.2.12	dpoint3d	146
9.1.2.13	float_2complex_image3df	146
9.1.2.14	int_u8_1complex_image2d	146
9.1.2.15	int_u8_2complex_image2d	146
9.1.2.16	int_u8_2complex_image3df	146
9.1.2.17	p_run2d	146
9.1.2.18	p_runs2d	147
9.1.2.19	point1d	147
9.1.2.20	point1df	147
9.1.2.21	point2d	147
9.1.2.22	point2d_h	147
9.1.2.23	point2df	147
9.1.2.24	point3d	147
9.1.2.25	point3df	147
9.1.2.26	rgb8_2complex_image3df	148
9.1.2.27	space_2complex_geometry	148
9.1.2.28	unsigned_2complex_image3df	148
9.1.2.29	vec2d_d	148
9.1.2.30	vec2d_f	148
9.1.2.31	vec3d_d	148
9.1.2.32	vec3d_f	148
9.1.3	Function Documentation	148
9.1.3.1	a_point_of	148

9.1.3.2	apply_p2p	149
9.1.3.3	apply_p2p	149
9.1.3.4	compose	149
9.1.3.5	duplicate	149
9.1.3.6	extend	150
9.1.3.7	extend	150
9.1.3.8	extend	150
9.1.3.9	implies	150
9.1.3.10	initialize	150
9.1.3.11	larger_than	151
9.1.3.12	make_debug_graph_image	151
9.1.3.13	mln_exact	151
9.1.3.14	mln_gen_complex_neighborhood	151
9.1.3.15	mln_gen_complex_neighborhood	151
9.1.3.16	mln_gen_complex_neighborhood	152
9.1.3.17	mln_gen_complex_neighborhood	152
9.1.3.18	mln_gen_complex_neighborhood	152
9.1.3.19	mln_gen_complex_neighborhood	152
9.1.3.20	mln_gen_complex_window	152
9.1.3.21	mln_gen_complex_window	152
9.1.3.22	mln_gen_complex_window	152
9.1.3.23	mln_gen_complex_window	152
9.1.3.24	mln_gen_complex_window	153
9.1.3.25	mln_gen_complex_window	153
9.1.3.26	mln_gen_complex_window_p	153
9.1.3.27	mln_gen_complex_window_p	153
9.1.3.28	mln_gen_complex_window_p	153
9.1.3.29	mln_gen_complex_window_p	153
9.1.3.30	mln_gen_complex_window_p	153
9.1.3.31	mln_gen_complex_window_p	153
9.1.3.32	mln_regular	154
9.1.3.33	mln_trait_op_geq	154
9.1.3.34	mln_trait_op_greater	154
9.1.3.35	mln_trait_op_leq	154
9.1.3.36	mln_trait_op_neq	154
9.1.3.37	operator!=	155

9.1.3.38	operator!=	155
9.1.3.39	operator*	155
9.1.3.40	operator++	156
9.1.3.41	operator-	156
9.1.3.42	operator-	156
9.1.3.43	operator--	156
9.1.3.44	operator<	156
9.1.3.45	operator<	157
9.1.3.46	operator<	157
9.1.3.47	operator<<	157
9.1.3.48	operator<<	157
9.1.3.49	operator<<	158
9.1.3.50	operator<<	158
9.1.3.51	operator<=	158
9.1.3.52	operator<=	158
9.1.3.53	operator<=	158
9.1.3.54	operator<=	159
9.1.3.55	operator<=	159
9.1.3.56	operator==	159
9.1.3.57	operator==	159
9.1.3.58	operator==	159
9.1.3.59	operator==	159
9.1.3.60	operator==	160
9.1.3.61	operator==	160
9.1.3.62	operator==	160
9.1.3.63	operator	160
9.1.3.64	operator	161
9.1.3.65	operator	161
9.1.3.66	operator	161
9.1.3.67	operator	161
9.1.3.68	operator	161
9.1.3.69	primary	162
9.1.3.70	ptransform	162
9.1.4	Variable Documentation	162
9.1.4.1	before	162
9.1.4.2	sagittal_dec	162

9.1.4.3	up	162
9.2	mln::accu Namespace Reference	162
9.2.1	Detailed Description	164
9.2.2	Function Documentation	164
9.2.2.1	compute	164
9.2.2.2	line	165
9.2.2.3	mln_meta_accu_result	165
9.2.2.4	take	166
9.3	mln::accu::image Namespace Reference	166
9.3.1	Detailed Description	166
9.4	mln::accu::impl Namespace Reference	166
9.4.1	Detailed Description	166
9.5	mln::accu::logic Namespace Reference	166
9.5.1	Detailed Description	167
9.6	mln::accu::math Namespace Reference	167
9.6.1	Detailed Description	167
9.7	mln::accu::meta::logic Namespace Reference	167
9.7.1	Detailed Description	168
9.8	mln::accu::meta::math Namespace Reference	168
9.8.1	Detailed Description	168
9.9	mln::accu::meta::shape Namespace Reference	168
9.9.1	Detailed Description	169
9.10	mln::accu::meta::stat Namespace Reference	169
9.10.1	Detailed Description	169
9.11	mln::accu::shape Namespace Reference	170
9.11.1	Detailed Description	170
9.12	mln::accu::stat Namespace Reference	170
9.12.1	Detailed Description	171
9.12.2	Function Documentation	171
9.12.2.1	operator==	171
9.13	mln::algebra Namespace Reference	172
9.13.1	Detailed Description	172
9.13.2	Function Documentation	173
9.13.2.1	ldlt_decomp	173
9.13.2.2	ldlt_solve	173
9.13.2.3	operator*	173

9.13.2.4	vprod	173
9.14	mln::arith Namespace Reference	173
9.14.1	Detailed Description	176
9.14.2	Function Documentation	176
9.14.2.1	diff_abs	176
9.14.2.2	div	176
9.14.2.3	div_cst	177
9.14.2.4	div_inplace	177
9.14.2.5	min	178
9.14.2.6	min_inplace	178
9.14.2.7	minus	178
9.14.2.8	minus	179
9.14.2.9	minus	179
9.14.2.10	minus_cst	179
9.14.2.11	minus_cst	180
9.14.2.12	minus_cst_inplace	180
9.14.2.13	minus_inplace	181
9.14.2.14	plus	181
9.14.2.15	plus	181
9.14.2.16	plus	182
9.14.2.17	plus_cst	182
9.14.2.18	plus_cst	183
9.14.2.19	plus_cst	183
9.14.2.20	plus_cst_inplace	183
9.14.2.21	plus_inplace	184
9.14.2.22	revert	184
9.14.2.23	revert_inplace	185
9.14.2.24	times	185
9.14.2.25	times_cst	185
9.14.2.26	times_inplace	186
9.15	mln::arith::impl Namespace Reference	186
9.15.1	Detailed Description	186
9.16	mln::arith::impl::generic Namespace Reference	186
9.16.1	Detailed Description	186
9.17	mln::binarization Namespace Reference	187
9.17.1	Detailed Description	187

9.17.2	Function Documentation	187
9.17.2.1	binarization	187
9.17.2.2	threshold	187
9.18	mln::border Namespace Reference	188
9.18.1	Detailed Description	188
9.18.2	Function Documentation	188
9.18.2.1	adjust	188
9.18.2.2	duplicate	189
9.18.2.3	equalize	189
9.18.2.4	fill	190
9.18.2.5	find	190
9.18.2.6	get	190
9.18.2.7	mirror	191
9.18.2.8	resize	191
9.19	mln::border::impl Namespace Reference	191
9.19.1	Detailed Description	191
9.20	mln::border::impl::generic Namespace Reference	192
9.20.1	Detailed Description	192
9.21	mln::canvas Namespace Reference	192
9.21.1	Detailed Description	193
9.21.2	Function Documentation	193
9.21.2.1	distance_front	193
9.21.2.2	distance_geodesic	193
9.22	mln::canvas::browsing Namespace Reference	193
9.22.1	Detailed Description	194
9.23	mln::canvas::impl Namespace Reference	194
9.23.1	Detailed Description	194
9.24	mln::canvas::labeling Namespace Reference	194
9.24.1	Detailed Description	195
9.24.2	Function Documentation	195
9.24.2.1	blobs	195
9.25	mln::canvas::labeling::impl Namespace Reference	195
9.25.1	Detailed Description	195
9.26	mln::canvas::morpho Namespace Reference	195
9.26.1	Detailed Description	195
9.27	mln::convert Namespace Reference	196

9.27.1 Detailed Description	198
9.27.2 Function Documentation	198
9.27.2.1 from_to	198
9.27.2.2 from_to	198
9.27.2.3 from_to	198
9.27.2.4 from_to	198
9.27.2.5 mln_image_from_grid	199
9.27.2.6 mln_image_from_grid	199
9.27.2.7 mln_image_from_grid	199
9.27.2.8 mln_image_from_grid	199
9.27.2.9 mln_window	199
9.27.2.10 to	199
9.27.2.11 to_dpoint	199
9.27.2.12 to_fun	199
9.27.2.13 to_image	200
9.27.2.14 to_p_array	200
9.27.2.15 to_p_array	200
9.27.2.16 to_p_array	200
9.27.2.17 to_p_set	200
9.27.2.18 to_p_set	200
9.27.2.19 to_p_set	200
9.27.2.20 to_p_set	201
9.27.2.21 to_p_set	201
9.27.2.22 to_qimage	201
9.27.2.23 to_upper_window	201
9.27.2.24 to_upper_window	201
9.27.2.25 to_window	201
9.27.2.26 to_window	202
9.27.2.27 to_window	202
9.27.3 Variable Documentation	202
9.27.3.1 to_fun	202
9.28 mln::data Namespace Reference	202
9.28.1 Detailed Description	204
9.28.2 Function Documentation	204
9.28.2.1 abs	204
9.28.2.2 abs_inplace	205

9.28.2.3	apply	205
9.28.2.4	compute	205
9.28.2.5	compute	206
9.28.2.6	convert	206
9.28.2.7	fast_median	206
9.28.2.8	fill	207
9.28.2.9	fill_with_image	207
9.28.2.10	fill_with_value	207
9.28.2.11	median	208
9.28.2.12	mln_meta_accu_result	208
9.28.2.13	paste	208
9.28.2.14	paste_without_localization	209
9.28.2.15	replace	209
9.28.2.16	saturate	209
9.28.2.17	saturate	210
9.28.2.18	saturate_inplace	210
9.28.2.19	sort_offsets_increasing	210
9.28.2.20	sort_psites_decreasing	210
9.28.2.21	sort_psites_increasing	211
9.28.2.22	stretch	211
9.28.2.23	to_enc	211
9.28.2.24	transform	212
9.28.2.25	transform	212
9.28.2.26	transform_inplace	212
9.28.2.27	transform_inplace	213
9.28.2.28	update	213
9.28.2.29	wrap	213
9.29	mln::data::approx Namespace Reference	214
9.29.1	Detailed Description	214
9.29.2	Function Documentation	214
9.29.2.1	median	214
9.29.2.2	median	214
9.29.2.3	median	215
9.30	mln::data::approx::impl Namespace Reference	215
9.30.1	Detailed Description	215
9.31	mln::data::impl Namespace Reference	215

9.31.1	Detailed Description	216
9.31.2	Function Documentation	216
9.31.2.1	paste_without_localization_fast	216
9.31.2.2	paste_without_localization_fastest	217
9.31.2.3	paste_without_localization_lines	217
9.31.2.4	stretch	218
9.31.2.5	transform_inplace_lowq	218
9.31.2.6	update_fastest	218
9.32	mln::data::impl::generic Namespace Reference	218
9.32.1	Detailed Description	219
9.32.2	Function Documentation	219
9.32.2.1	fill_with_image	219
9.32.2.2	fill_with_value	220
9.32.2.3	paste	220
9.32.2.4	transform	220
9.32.2.5	transform	220
9.32.2.6	transform_inplace	221
9.32.2.7	transform_inplace	221
9.32.2.8	update	221
9.33	mln::data::naive Namespace Reference	221
9.33.1	Detailed Description	222
9.33.2	Function Documentation	222
9.33.2.1	median	222
9.34	mln::data::naive::impl Namespace Reference	222
9.34.1	Detailed Description	222
9.35	mln::debug Namespace Reference	223
9.35.1	Detailed Description	224
9.35.2	Function Documentation	224
9.35.2.1	draw_graph	224
9.35.2.2	draw_graph	225
9.35.2.3	draw_graph	225
9.35.2.4	filename	225
9.35.2.5	format	225
9.35.2.6	format	226
9.35.2.7	format	226
9.35.2.8	format	226

9.35.2.9	iota	226
9.35.2.10	mosaic	226
9.35.2.11	println	227
9.35.2.12	println	227
9.35.2.13	println_with_border	227
9.35.2.14	put_word	227
9.35.2.15	slices_2d	227
9.35.2.16	slices_2d	227
9.35.2.17	superpose	228
9.35.2.18	superpose	228
9.35.2.19	z_order	228
9.36	mln::debug::impl Namespace Reference	228
9.36.1	Detailed Description	229
9.37	mln::def Namespace Reference	229
9.37.1	Detailed Description	229
9.37.2	Typedef Documentation	229
9.37.2.1	coord	229
9.37.2.2	coordf	229
9.37.3	Enumeration Type Documentation	229
9.37.3.1	"@21	229
9.38	mln::display Namespace Reference	230
9.38.1	Detailed Description	230
9.39	mln::display::impl Namespace Reference	230
9.39.1	Detailed Description	230
9.40	mln::display::impl::generic Namespace Reference	230
9.40.1	Detailed Description	230
9.41	mln::doc Namespace Reference	230
9.41.1	Detailed Description	232
9.42	mln::draw Namespace Reference	232
9.42.1	Detailed Description	232
9.42.2	Function Documentation	232
9.42.2.1	box	232
9.42.2.2	box_plain	233
9.42.2.3	dashed_line	233
9.42.2.4	line	234
9.42.2.5	plot	234

9.42.2.6	polygon	235
9.42.2.7	site_set	235
9.43	mln::estim Namespace Reference	235
9.43.1	Detailed Description	236
9.43.2	Function Documentation	236
9.43.2.1	mean	236
9.43.2.2	mean	236
9.43.2.3	min_max	237
9.43.2.4	sum	237
9.43.2.5	sum	237
9.44	mln::extension Namespace Reference	237
9.44.1	Detailed Description	238
9.44.2	Function Documentation	238
9.44.2.1	adjust	238
9.44.2.2	adjust	239
9.44.2.3	adjust	239
9.44.2.4	adjust	239
9.44.2.5	adjust_duplicate	239
9.44.2.6	adjust_fill	239
9.44.2.7	duplicate	239
9.44.2.8	fill	240
9.45	mln::fun Namespace Reference	240
9.45.1	Detailed Description	241
9.46	mln::fun::access Namespace Reference	241
9.46.1	Detailed Description	241
9.47	mln::fun::i2v Namespace Reference	242
9.47.1	Detailed Description	242
9.47.2	Function Documentation	242
9.47.2.1	operator<<	242
9.48	mln::fun::n2v Namespace Reference	242
9.48.1	Detailed Description	242
9.49	mln::fun::p2b Namespace Reference	242
9.49.1	Detailed Description	243
9.50	mln::fun::p2p Namespace Reference	243
9.50.1	Detailed Description	243
9.51	mln::fun::p2v Namespace Reference	243

9.51.1 Detailed Description	243
9.52 mln::fun::stat Namespace Reference	243
9.52.1 Detailed Description	243
9.53 mln::fun::v2b Namespace Reference	243
9.53.1 Detailed Description	244
9.54 mln::fun::v2i Namespace Reference	244
9.54.1 Detailed Description	244
9.55 mln::fun::v2v Namespace Reference	244
9.55.1 Detailed Description	245
9.55.2 Variable Documentation	245
9.55.2.1 f_hsi_to_rgb_3x8	245
9.55.2.2 f_hsl_to_rgb_3x8	245
9.55.2.3 f_rgb_to_hsi_f	245
9.55.2.4 f_rgb_to_hsl_f	245
9.56 mln::fun::v2w2v Namespace Reference	246
9.56.1 Detailed Description	246
9.57 mln::fun::v2w_w2v Namespace Reference	246
9.57.1 Detailed Description	246
9.58 mln::fun::vv2b Namespace Reference	246
9.58.1 Detailed Description	247
9.59 mln::fun::vv2v Namespace Reference	247
9.59.1 Detailed Description	248
9.60 mln::fun::x2p Namespace Reference	248
9.60.1 Detailed Description	248
9.61 mln::fun::x2v Namespace Reference	248
9.61.1 Detailed Description	248
9.62 mln::fun::x2x Namespace Reference	248
9.62.1 Detailed Description	249
9.63 mln::geom Namespace Reference	249
9.63.1 Detailed Description	253
9.63.2 Function Documentation	253
9.63.2.1 bbox	253
9.63.2.2 bbox	253
9.63.2.3 bbox	253
9.63.2.4 bbox	254
9.63.2.5 chamfer	254

9.63.2.6	delta	254
9.63.2.7	delta	254
9.63.2.8	delta	254
9.63.2.9	max_col	254
9.63.2.10	max_col	255
9.63.2.11	max_ind	255
9.63.2.12	max_row	255
9.63.2.13	max_row	255
9.63.2.14	max_sli	255
9.63.2.15	mesh_corner_point_area	255
9.63.2.16	mesh_curvature	256
9.63.2.17	mesh_normal	256
9.63.2.18	min_col	257
9.63.2.19	min_col	257
9.63.2.20	min_ind	257
9.63.2.21	min_row	257
9.63.2.22	min_row	257
9.63.2.23	min_sli	257
9.63.2.24	ncols	258
9.63.2.25	ncols	258
9.63.2.26	ninds	258
9.63.2.27	nrows	258
9.63.2.28	nrows	258
9.63.2.29	nsites	258
9.63.2.30	nslis	259
9.63.2.31	pmin_pmax	259
9.63.2.32	pmin_pmax	259
9.63.2.33	pmin_pmax	259
9.63.2.34	pmin_pmax	259
9.63.2.35	rotate	259
9.63.2.36	rotate	260
9.63.2.37	rotate	260
9.63.2.38	rotate	260
9.63.2.39	rotate	260
9.63.2.40	seeds2tiling	261
9.63.2.41	seeds2tiling_roundness	261

9.63.2.42	translate	261
9.63.2.43	translate	262
9.63.2.44	translate	262
9.63.2.45	vertical_symmetry	262
9.64	mln::geom::impl Namespace Reference	262
9.64.1	Detailed Description	263
9.64.2	Function Documentation	263
9.64.2.1	seeds2tiling	263
9.65	mln::graph Namespace Reference	263
9.65.1	Detailed Description	264
9.65.2	Function Documentation	264
9.65.2.1	compute	264
9.65.2.2	labeling	264
9.65.2.3	to_neighb	265
9.65.2.4	to_win	265
9.66	mln::grid Namespace Reference	265
9.66.1	Detailed Description	265
9.67	mln::histo Namespace Reference	266
9.67.1	Detailed Description	266
9.67.2	Function Documentation	266
9.67.2.1	compute	266
9.67.2.2	equalize	266
9.68	mln::histo::impl Namespace Reference	267
9.68.1	Detailed Description	267
9.69	mln::histo::impl::generic Namespace Reference	267
9.69.1	Detailed Description	267
9.70	mln::impl Namespace Reference	267
9.70.1	Detailed Description	267
9.71	mln::io Namespace Reference	267
9.71.1	Detailed Description	269
9.72	mln::io::cloud Namespace Reference	269
9.72.1	Detailed Description	269
9.72.2	Function Documentation	269
9.72.2.1	load	269
9.72.2.2	save	270
9.73	mln::io::dicom Namespace Reference	270

9.73.1 Detailed Description	270
9.73.2 Function Documentation	270
9.73.2.1 get_header	270
9.73.2.2 load	270
9.74 mln::io::dump Namespace Reference	271
9.74.1 Detailed Description	271
9.74.2 Function Documentation	271
9.74.2.1 get_header	271
9.74.2.2 load	272
9.74.2.3 save	272
9.75 mln::io::fits Namespace Reference	272
9.75.1 Detailed Description	272
9.75.2 Function Documentation	272
9.75.2.1 load	272
9.75.2.2 load	273
9.76 mln::io::fld Namespace Reference	273
9.76.1 Detailed Description	273
9.76.2 Function Documentation	274
9.76.2.1 load	274
9.76.2.2 read_header	274
9.76.2.3 write_header	274
9.77 mln::io::magick Namespace Reference	274
9.77.1 Detailed Description	275
9.77.2 Function Documentation	275
9.77.2.1 load	275
9.77.2.2 save	275
9.77.2.3 save	276
9.78 mln::io::off Namespace Reference	276
9.78.1 Detailed Description	276
9.78.2 Function Documentation	276
9.78.2.1 load	276
9.78.2.2 save	277
9.78.2.3 save_bin_alt	277
9.79 mln::io::pbm Namespace Reference	277
9.79.1 Detailed Description	278
9.79.2 Function Documentation	278

9.79.2.1	load	278
9.79.2.2	load	278
9.79.2.3	save	279
9.80	mln::io::pbm::impl Namespace Reference	279
9.80.1	Detailed Description	279
9.81	mln::io::pbms Namespace Reference	279
9.81.1	Detailed Description	279
9.81.2	Function Documentation	279
9.81.2.1	load	279
9.82	mln::io::pbms::impl Namespace Reference	280
9.82.1	Detailed Description	280
9.83	mln::io::pfm Namespace Reference	280
9.83.1	Detailed Description	280
9.83.2	Function Documentation	281
9.83.2.1	load	281
9.83.2.2	load	281
9.83.2.3	save	281
9.84	mln::io::pfm::impl Namespace Reference	281
9.84.1	Detailed Description	281
9.85	mln::io::pgm Namespace Reference	282
9.85.1	Detailed Description	282
9.85.2	Function Documentation	282
9.85.2.1	load	282
9.85.2.2	load	282
9.85.2.3	save	283
9.86	mln::io::pgms Namespace Reference	283
9.86.1	Detailed Description	283
9.86.2	Function Documentation	283
9.86.2.1	load	283
9.87	mln::io::plot Namespace Reference	283
9.87.1	Detailed Description	284
9.87.2	Function Documentation	284
9.87.2.1	load	284
9.87.2.2	save	284
9.87.2.3	save	285
9.87.2.4	save	285

9.88	mln::io::pnm Namespace Reference	285
9.88.1	Detailed Description	286
9.88.2	Function Documentation	286
9.88.2.1	load	286
9.88.2.2	load	286
9.88.2.3	load_raw_2d	286
9.88.2.4	max_component	286
9.88.2.5	save	287
9.89	mln::io::pnm::impl Namespace Reference	287
9.89.1	Detailed Description	287
9.90	mln::io::pnms Namespace Reference	287
9.90.1	Detailed Description	287
9.90.2	Function Documentation	287
9.90.2.1	load	287
9.90.2.2	load	288
9.91	mln::io::ppm Namespace Reference	288
9.91.1	Detailed Description	288
9.91.2	Function Documentation	288
9.91.2.1	load	288
9.91.2.2	load	289
9.91.2.3	save	289
9.92	mln::io::ppms Namespace Reference	289
9.92.1	Detailed Description	290
9.92.2	Function Documentation	290
9.92.2.1	load	290
9.93	mln::io::raw Namespace Reference	290
9.93.1	Detailed Description	290
9.93.2	Function Documentation	291
9.93.2.1	get_header	291
9.93.2.2	load	291
9.93.2.3	save	291
9.94	mln::io::tiff Namespace Reference	291
9.94.1	Detailed Description	292
9.94.2	Function Documentation	292
9.94.2.1	load	292
9.95	mln::io::txt Namespace Reference	292

9.95.1 Detailed Description	292
9.95.2 Function Documentation	292
9.95.2.1 save	292
9.96 mln::labeling Namespace Reference	292
9.96.1 Detailed Description	295
9.96.2 Function Documentation	296
9.96.2.1 background	296
9.96.2.2 blobs	296
9.96.2.3 blobs_and_compute	297
9.96.2.4 colorize	297
9.96.2.5 colorize	297
9.96.2.6 colorize	298
9.96.2.7 compute	298
9.96.2.8 compute	298
9.96.2.9 compute	299
9.96.2.10 compute	299
9.96.2.11 compute	299
9.96.2.12 compute_image	300
9.96.2.13 compute_image	300
9.96.2.14 compute_image	301
9.96.2.15 fill_holes	301
9.96.2.16 flat_zones	302
9.96.2.17 foreground	302
9.96.2.18 pack	302
9.96.2.19 pack	303
9.96.2.20 pack_inplace	303
9.96.2.21 pack_inplace	303
9.96.2.22 regional_maxima	304
9.96.2.23 regional_minima	304
9.96.2.24 relabel	304
9.96.2.25 relabel	305
9.96.2.26 relabel_inplace	305
9.96.2.27 relabel_inplace	305
9.96.2.28 superpose	306
9.96.2.29 value	306
9.96.2.30 value_and_compute	307

9.96.2.31	wrap	307
9.96.2.32	wrap	307
9.97	mln::labeling::impl Namespace Reference	308
9.97.1	Detailed Description	308
9.97.2	Function Documentation	308
9.97.2.1	compute_fastest	308
9.97.2.2	compute_fastest	309
9.98	mln::labeling::impl::generic Namespace Reference	309
9.98.1	Detailed Description	310
9.98.2	Function Documentation	310
9.98.2.1	compute	310
9.98.2.2	compute	310
9.98.2.3	compute	311
9.98.2.4	compute	311
9.99	mln::linear Namespace Reference	311
9.99.1	Detailed Description	312
9.99.2	Function Documentation	312
9.99.2.1	gaussian	312
9.99.2.2	gaussian	313
9.99.2.3	gaussian_1st_derivative	313
9.99.2.4	gaussian_1st_derivative	313
9.99.2.5	gaussian_2nd_derivative	314
9.99.2.6	gaussian_2nd_derivative	314
9.99.2.7	mln_ch_convolve	314
9.99.2.8	mln_ch_convolve	314
9.99.2.9	mln_ch_convolve	315
9.99.2.10	mln_ch_convolve	315
9.99.2.11	mln_ch_convolve_grad	315
9.100	mln::linear::impl Namespace Reference	315
9.100.1	Detailed Description	315
9.101	mln::linear::local Namespace Reference	315
9.101.1	Detailed Description	316
9.101.2	Function Documentation	316
9.101.2.1	convolve	316
9.101.2.2	convolve	316
9.102	mln::linear::local::impl Namespace Reference	317

9.102.1 Detailed Description	317
9.103mln::literal Namespace Reference	317
9.103.1 Detailed Description	320
9.103.2 Variable Documentation	320
9.103.2.1 black	320
9.103.2.2 blue	320
9.103.2.3 brown	320
9.103.2.4 cyan	320
9.103.2.5 dark_gray	320
9.103.2.6 green	321
9.103.2.7 identity	321
9.103.2.8 light_gray	321
9.103.2.9 lime	321
9.103.2.10magenta	321
9.103.2.11lmax	321
9.103.2.12medium_gray	321
9.103.2.13min	321
9.103.2.14olive	322
9.103.2.15one	322
9.103.2.16orange	322
9.103.2.17origin	322
9.103.2.18pink	322
9.103.2.19purple	322
9.103.2.20red	322
9.103.2.21teal	322
9.103.2.22violet	323
9.103.2.23white	323
9.103.2.24yellow	323
9.103.2.25zero	323
9.104mln::logical Namespace Reference	323
9.104.1 Detailed Description	324
9.104.2 Function Documentation	324
9.104.2.1 and_inplace	324
9.104.2.2 and_not	324
9.104.2.3 and_not_inplace	325
9.104.2.4 not_inplace	325

9.104.2.5 or_inplace	326
9.104.2.6 xor_inplace	326
9.105mln::logical::impl Namespace Reference	326
9.105.1 Detailed Description	327
9.106mln::logical::impl::generic Namespace Reference	327
9.106.1 Detailed Description	327
9.107mln::make Namespace Reference	327
9.107.1 Detailed Description	332
9.107.2 Function Documentation	332
9.107.2.1 attachment	332
9.107.2.2 box1d	332
9.107.2.3 box1d	333
9.107.2.4 box2d	333
9.107.2.5 box2d	333
9.107.2.6 box2d_h	334
9.107.2.7 box2d_h	334
9.107.2.8 box3d	335
9.107.2.9 box3d	335
9.107.2.10cell	336
9.107.2.11couple	336
9.107.2.12detachment	336
9.107.2.13dpoint2d_h	336
9.107.2.14dummy_p_edges	337
9.107.2.15dummy_p_edges	337
9.107.2.16dummy_p_vertices	337
9.107.2.17dummy_p_vertices	338
9.107.2.18edge_image	338
9.107.2.19edge_image	338
9.107.2.20edge_image	339
9.107.2.21edge_image	339
9.107.2.22edge_image	339
9.107.2.23edge_image	340
9.107.2.24h_mat	340
9.107.2.25image	340
9.107.2.26image	340
9.107.2.27image	341

9.107.2.28	image2d	341
9.107.2.29	image3d	341
9.107.2.30	image3d	341
9.107.2.31	influence_zone_adjacency_graph	342
9.107.2.32	mat	342
9.107.2.33	ord_pair	342
9.107.2.34	p_edges_with_mass_centers	342
9.107.2.35	p_vertices_with_mass_centers	343
9.107.2.36	pix	343
9.107.2.37	pixel	343
9.107.2.38	pixel	344
9.107.2.39	point2d_h	344
9.107.2.40	rag_and_labeled_wsl	344
9.107.2.41	region_adjacency_graph	345
9.107.2.42	relabelfun	345
9.107.2.43	relabelfun	346
9.107.2.44	vec	346
9.107.2.45	vec	346
9.107.2.46	vec	347
9.107.2.47	vec	347
9.107.2.48	vertex_image	347
9.107.2.49	vertex_image	348
9.107.2.50	voronoi	348
9.107.2.51	w_window	348
9.107.2.52	w_window1d	349
9.107.2.53	w_window2d	349
9.107.2.54	w_window3d	349
9.107.2.55	w_window_directional	350
9.108	mln::math Namespace Reference	350
9.108.1	Detailed Description	350
9.108.2	Function Documentation	350
9.108.2.1	abs	350
9.108.2.2	abs	351
9.108.2.3	abs	351
9.109	mln::metal Namespace Reference	351
9.109.1	Detailed Description	352

9.110mln::metal::impl Namespace Reference	352
9.110.1 Detailed Description	352
9.111mln::metal::math Namespace Reference	352
9.111.1 Detailed Description	352
9.112mln::metal::math::impl Namespace Reference	352
9.112.1 Detailed Description	352
9.113mln::morpho Namespace Reference	353
9.113.1 Detailed Description	356
9.113.2 Function Documentation	356
9.113.2.1 complementation	356
9.113.2.2 complementation_inplace	356
9.113.2.3 contrast	356
9.113.2.4 dilation	356
9.113.2.5 erosion	357
9.113.2.6 erosion_tolerant	357
9.113.2.7 general	357
9.113.2.8 gradient	357
9.113.2.9 gradient_external	357
9.113.2.10gradient_internal	358
9.113.2.11hit_or_miss	358
9.113.2.12hit_or_miss_background_closing	358
9.113.2.13hit_or_miss_background_opening	358
9.113.2.14hit_or_miss_closing	358
9.113.2.15hit_or_miss_opening	359
9.113.2.16laplacian	359
9.113.2.17line_gradient	359
9.113.2.18meyer_wst	359
9.113.2.19meyer_wst	360
9.113.2.20min	360
9.113.2.21min_inplace	360
9.113.2.22minus	360
9.113.2.23plus	361
9.113.2.24rank_filter	361
9.113.2.25thick_miss	361
9.113.2.26thickening	361
9.113.2.27thin_fit	361

9.113.2.2	thinning	362
9.113.2.2.9	top_hat_black	362
9.113.2.3	top_hat_self_complementary	362
9.113.2.3.1	top_hat_white	362
9.114	mln::morpho::approx Namespace Reference	363
9.114.1	Detailed Description	363
9.115	mln::morpho::attribute Namespace Reference	363
9.115.1	Detailed Description	363
9.116	mln::morpho::closing::approx Namespace Reference	363
9.116.1	Detailed Description	364
9.116.2	Function Documentation	364
9.116.2.1	structural	364
9.117	mln::morpho::elementary Namespace Reference	364
9.117.1	Detailed Description	365
9.117.2	Function Documentation	365
9.117.2.1	closing	365
9.117.2.2	mln_trait_op_minus_twice	365
9.117.2.3	opening	365
9.117.2.4	top_hat_black	365
9.117.2.5	top_hat_self_complementary	366
9.117.2.6	top_hat_white	366
9.118	mln::morpho::impl Namespace Reference	366
9.118.1	Detailed Description	366
9.119	mln::morpho::impl::generic Namespace Reference	366
9.119.1	Detailed Description	366
9.120	mln::morpho::opening::approx Namespace Reference	367
9.120.1	Detailed Description	367
9.120.2	Function Documentation	367
9.120.2.1	structural	367
9.121	mln::morpho::reconstruction Namespace Reference	367
9.121.1	Detailed Description	367
9.122	mln::morpho::reconstruction::by_dilation Namespace Reference	368
9.122.1	Detailed Description	368
9.123	mln::morpho::reconstruction::by_erosion Namespace Reference	368
9.123.1	Detailed Description	368
9.124	mln::morpho::tree Namespace Reference	368

9.124.1 Detailed Description	369
9.124.2 Function Documentation	370
9.124.2.1 compute_attribute_image	370
9.124.2.2 compute_attribute_image_from	370
9.124.2.3 compute_parent	371
9.124.2.4 dual_input_max_tree	372
9.124.2.5 max_tree	372
9.124.2.6 min_tree	372
9.124.2.7 propagate_if	373
9.124.2.8 propagate_if	373
9.124.2.9 propagate_if_value	373
9.124.2.10 propagate_if_value	374
9.124.2.11 propagate_node_to_ancestors	374
9.124.2.12 propagate_node_to_ancestors	374
9.124.2.13 propagate_node_to_descendants	375
9.124.2.14 propagate_node_to_descendants	375
9.124.2.15 propagate_representative	375
9.125 mln::morpho::tree::filter Namespace Reference	375
9.125.1 Detailed Description	376
9.125.2 Function Documentation	376
9.125.2.1 direct	376
9.125.2.2 filter	376
9.125.2.3 max	377
9.125.2.4 min	377
9.125.2.5 subtractive	377
9.126 mln::morpho::watershed Namespace Reference	378
9.126.1 Detailed Description	378
9.126.2 Function Documentation	379
9.126.2.1 flooding	379
9.126.2.2 flooding	379
9.126.2.3 superpose	379
9.126.2.4 superpose	380
9.126.2.5 topological	380
9.127 mln::morpho::watershed::watershed Namespace Reference	380
9.127.1 Detailed Description	380
9.128 mln::morpho::watershed::watershed::generic Namespace Reference	380

9.128.1 Detailed Description	380
9.129mln::norm Namespace Reference	381
9.129.1 Detailed Description	382
9.129.2 Function Documentation	382
9.129.2.1 l1	382
9.129.2.2 l1_distance	382
9.129.2.3 l2	382
9.129.2.4 l2_distance	382
9.129.2.5 linfty	382
9.129.2.6 linfty_distance	382
9.129.2.7 sqr_l2	383
9.130mln::norm::impl Namespace Reference	383
9.130.1 Detailed Description	383
9.131mln::opt Namespace Reference	383
9.131.1 Detailed Description	384
9.131.2 Function Documentation	384
9.131.2.1 at	384
9.131.2.2 at	384
9.131.2.3 at	384
9.131.2.4 at	384
9.131.2.5 at	384
9.131.2.6 at	385
9.132mln::opt::impl Namespace Reference	385
9.132.1 Detailed Description	385
9.133mln::pw Namespace Reference	385
9.133.1 Detailed Description	385
9.134mln::registration Namespace Reference	385
9.134.1 Detailed Description	386
9.134.2 Function Documentation	386
9.134.2.1 get_rot	386
9.134.2.2 icp	387
9.134.2.3 icp	387
9.134.2.4 registration1	388
9.134.2.5 registration2	388
9.134.2.6 registration3	388
9.135mln::select Namespace Reference	388

9.135.1 Detailed Description	388
9.136mln::set Namespace Reference	388
9.136.1 Detailed Description	389
9.136.2 Function Documentation	389
9.136.2.1 card	389
9.136.2.2 compute	390
9.136.2.3 compute_with_weights	390
9.136.2.4 compute_with_weights	390
9.136.2.5 get	391
9.136.2.6 has	391
9.136.2.7 mln_meta_accu_result	391
9.136.2.8 mln_meta_accu_result	391
9.137mln::subsampling Namespace Reference	391
9.137.1 Detailed Description	392
9.137.2 Function Documentation	392
9.137.2.1 antialiased	392
9.137.2.2 antialiased	392
9.137.2.3 gaussian_subsampling	392
9.137.2.4 subsampling	393
9.138mln::tag Namespace Reference	393
9.138.1 Detailed Description	393
9.139mln::test Namespace Reference	393
9.139.1 Detailed Description	394
9.139.2 Function Documentation	394
9.139.2.1 positive	394
9.139.2.2 predicate	394
9.139.2.3 predicate	394
9.139.2.4 predicate	394
9.140mln::test::impl Namespace Reference	395
9.140.1 Detailed Description	395
9.141mln::topo Namespace Reference	395
9.141.1 Detailed Description	399
9.141.2 Function Documentation	399
9.141.2.1 detach	399
9.141.2.2 edge	400
9.141.2.3 is_facet	400

9.141.2.4	make_algebraic_face	400
9.141.2.5	make_algebraic_n_face	400
9.141.2.6	operator!=	401
9.141.2.7	operator!=	401
9.141.2.8	operator!=	401
9.141.2.9	operator!=	401
9.141.2.10	operator+	402
9.141.2.11	operator-	402
9.141.2.12	operator-	402
9.141.2.13	operator-	402
9.141.2.14	operator<	402
9.141.2.15	operator<	402
9.141.2.16	operator<	403
9.141.2.17	operator<	403
9.141.2.18	operator<<	403
9.141.2.19	operator<<	403
9.141.2.20	operator<<	403
9.141.2.21	operator<<	404
9.141.2.22	operator<<	404
9.141.2.23	operator==	404
9.141.2.24	operator==	404
9.141.2.25	operator==	404
9.141.2.26	operator==	405
9.141.2.27	operator==	405
9.142	mln::trace Namespace Reference	405
9.142.1	Detailed Description	405
9.143	mln::trait Namespace Reference	405
9.143.1	Detailed Description	405
9.144	mln::transform Namespace Reference	405
9.144.1	Detailed Description	407
9.144.2	Function Documentation	407
9.144.2.1	distance_and_closest_point_geodesic	407
9.144.2.2	distance_and_closest_point_geodesic	407
9.144.2.3	distance_and_influence_zone_geodesic	408
9.144.2.4	distance_front	408
9.144.2.5	distance_geodesic	408

9.144.2.6	hough	409
9.144.2.7	influence_zone_front	409
9.144.2.8	influence_zone_front	409
9.144.2.9	influence_zone_geodesic	409
9.144.2.10	influence_zone_geodesic_saturated	410
9.144.2.11	influence_zone_geodesic_saturated	410
9.145	mln::util Namespace Reference	410
9.145.1	Detailed Description	414
9.145.2	Typedef Documentation	414
9.145.2.1	vertex_id_t	414
9.145.3	Function Documentation	414
9.145.3.1	display_branch	414
9.145.3.2	display_tree	414
9.145.3.3	lemmings	415
9.145.3.4	make_greater_point	415
9.145.3.5	make_greater_psite	415
9.145.3.6	operator<	415
9.145.3.7	operator<<	415
9.145.3.8	operator<<	415
9.145.3.9	operator==	416
9.145.3.10	operator==	416
9.145.3.11	ord_strict	416
9.145.3.12	ord_weak	416
9.145.3.13	tree_fast_to_image	416
9.145.3.14	tree_to_fast	416
9.145.3.15	tree_to_image	417
9.146	mln::util::impl Namespace Reference	417
9.146.1	Detailed Description	417
9.147	mln::value Namespace Reference	417
9.147.1	Detailed Description	422
9.147.2	Typedef Documentation	422
9.147.2.1	float01_16	422
9.147.2.2	float01_8	422
9.147.2.3	gl16	422
9.147.2.4	gl8	422
9.147.2.5	glf	422

9.147.2.6 int_s16	423
9.147.2.7 int_s32	423
9.147.2.8 int_s8	423
9.147.2.9 int_u12	423
9.147.2.10 int_u16	423
9.147.2.11 int_u32	423
9.147.2.12 int_u8	423
9.147.2.13 label_16	423
9.147.2.14 label_32	423
9.147.2.15 label_8	424
9.147.2.16 rgb16	424
9.147.2.17 rgb8	424
9.147.3 Function Documentation	424
9.147.3.1 cast	424
9.147.3.2 equiv	424
9.147.3.3 operator*	424
9.147.3.4 operator*	424
9.147.3.5 operator+	425
9.147.3.6 operator+	425
9.147.3.7 operator-	425
9.147.3.8 operator-	425
9.147.3.9 operator/	425
9.147.3.10 operator/	425
9.147.3.11 operator<<	425
9.147.3.12 operator<<	426
9.147.3.13 operator<<	426
9.147.3.14 operator<<	426
9.147.3.15 operator<<	426
9.147.3.16 operator<<	427
9.147.3.17 operator<<	427
9.147.3.18 operator<<	427
9.147.3.19 operator<<	428
9.147.3.20 operator<<	428
9.147.3.21 operator<<	428
9.147.3.22 operator==	428
9.147.3.23 operator==	429

9.147.3.24	other	429
9.147.3.25	stack	429
9.148	mln::value::impl Namespace Reference	429
9.148.1	Detailed Description	429
9.149	mln::win Namespace Reference	429
9.149.1	Detailed Description	431
9.149.2	Function Documentation	431
9.149.2.1	diff	431
9.149.2.2	mln_regular	431
9.149.2.3	mln_regular	431
9.149.2.4	sym	431
9.149.2.5	sym	431
10	Class Documentation	433
10.1	mln::accu::center< P, V > Struct Template Reference	433
10.1.1	Detailed Description	433
10.1.2	Member Function Documentation	434
10.1.2.1	init	434
10.1.2.2	is_valid	434
10.1.2.3	nsites	434
10.1.2.4	take_as_init	434
10.1.2.5	take_n_times	434
10.1.2.6	to_result	434
10.2	mln::accu::convolve< T1, T2, R > Struct Template Reference	435
10.2.1	Detailed Description	435
10.2.2	Member Function Documentation	435
10.2.2.1	init	435
10.2.2.2	is_valid	436
10.2.2.3	take_as_init	436
10.2.2.4	take_n_times	436
10.2.2.5	to_result	436
10.3	mln::accu::count_adjacent_vertices< F, S > Struct Template Reference	436
10.3.1	Detailed Description	437
10.3.2	Member Function Documentation	437
10.3.2.1	init	437
10.3.2.2	is_valid	437
10.3.2.3	set_value	437

10.3.2.4	take_as_init	437
10.3.2.5	take_n_times	438
10.3.2.6	to_result	438
10.4	mln::accu::count_labels< L > Struct Template Reference	438
10.4.1	Detailed Description	438
10.4.2	Member Function Documentation	439
10.4.2.1	init	439
10.4.2.2	is_valid	439
10.4.2.3	set_value	439
10.4.2.4	take_as_init	439
10.4.2.5	take_n_times	439
10.4.2.6	to_result	439
10.5	mln::accu::count_value< V > Struct Template Reference	439
10.5.1	Detailed Description	440
10.5.2	Member Function Documentation	440
10.5.2.1	init	440
10.5.2.2	is_valid	440
10.5.2.3	set_value	440
10.5.2.4	take_as_init	441
10.5.2.5	take_n_times	441
10.5.2.6	to_result	441
10.6	mln::accu::histo< V > Struct Template Reference	441
10.6.1	Detailed Description	442
10.6.2	Member Function Documentation	442
10.6.2.1	is_valid	442
10.6.2.2	take	442
10.6.2.3	take_as_init	442
10.6.2.4	take_n_times	442
10.6.2.5	vect	442
10.7	mln::accu::label_used< L > Struct Template Reference	442
10.7.1	Detailed Description	443
10.7.2	Member Function Documentation	443
10.7.2.1	init	443
10.7.2.2	is_valid	443
10.7.2.3	take	443
10.7.2.4	take_as_init	444

10.7.2.5	take_n_times	444
10.7.2.6	to_result	444
10.8	mln::accu::logic::land Struct Reference	444
10.8.1	Detailed Description	445
10.8.2	Member Function Documentation	445
10.8.2.1	init	445
10.8.2.2	is_valid	445
10.8.2.3	take_as_init	445
10.8.2.4	take_n_times	445
10.8.2.5	to_result	445
10.9	mln::accu::logic::land_basic Struct Reference	445
10.9.1	Detailed Description	446
10.9.2	Member Function Documentation	446
10.9.2.1	can_stop	446
10.9.2.2	init	446
10.9.2.3	is_valid	446
10.9.2.4	take_as_init	447
10.9.2.5	take_n_times	447
10.9.2.6	to_result	447
10.10	mln::accu::logic::lor Struct Reference	447
10.10.1	Detailed Description	447
10.10.2	Member Function Documentation	448
10.10.2.1	init	448
10.10.2.2	is_valid	448
10.10.2.3	take_as_init	448
10.10.2.4	take_n_times	448
10.10.2.5	to_result	448
10.11	mln::accu::logic::lor_basic Struct Reference	448
10.11.1	Detailed Description	449
10.11.2	Member Function Documentation	449
10.11.2.1	can_stop	449
10.11.2.2	init	449
10.11.2.3	is_valid	449
10.11.2.4	take_as_init	449
10.11.2.5	take_n_times	450
10.11.2.6	to_result	450

10.12	mln::accu::maj_h< T > Struct Template Reference	450
10.12.1	Detailed Description	450
10.12.2	Member Function Documentation	451
10.12.2.1	init	451
10.12.2.2	is_valid	451
10.12.2.3	take_as_init	451
10.12.2.4	take_n_times	451
10.12.2.5	to_result	451
10.13	mln::accu::math::count< T > Struct Template Reference	451
10.13.1	Detailed Description	452
10.13.2	Member Function Documentation	452
10.13.2.1	init	452
10.13.2.2	is_valid	452
10.13.2.3	set_value	452
10.13.2.4	take_as_init	452
10.13.2.5	take_n_times	453
10.13.2.6	to_result	453
10.14	mln::accu::math::inf< T > Struct Template Reference	453
10.14.1	Detailed Description	453
10.14.2	Member Function Documentation	454
10.14.2.1	init	454
10.14.2.2	is_valid	454
10.14.2.3	take_as_init	454
10.14.2.4	take_n_times	454
10.14.2.5	to_result	454
10.15	mln::accu::math::sum< T, S > Struct Template Reference	454
10.15.1	Detailed Description	455
10.15.2	Member Function Documentation	455
10.15.2.1	init	455
10.15.2.2	is_valid	455
10.15.2.3	take_as_init	455
10.15.2.4	take_n_times	455
10.15.2.5	to_result	456
10.16	mln::accu::math::sup< T > Struct Template Reference	456
10.16.1	Detailed Description	456
10.16.2	Member Function Documentation	456

10.16.2.1	init	456
10.16.2.2	is_valid	457
10.16.2.3	take_as_init	457
10.16.2.4	take_n_times	457
10.16.2.5	to_result	457
10.17	mln::accu::max_site< I > Struct Template Reference	457
10.17.1	Detailed Description	458
10.17.2	Member Function Documentation	458
10.17.2.1	init	458
10.17.2.2	is_valid	458
10.17.2.3	take_as_init	458
10.17.2.4	take_n_times	458
10.17.2.5	to_result	458
10.18	mln::accu::meta::center Struct Reference	459
10.18.1	Detailed Description	459
10.19	mln::accu::meta::count_adjacent_vertices Struct Reference	459
10.19.1	Detailed Description	460
10.20	mln::accu::meta::count_labels Struct Reference	460
10.20.1	Detailed Description	461
10.21	mln::accu::meta::count_value Struct Reference	461
10.21.1	Detailed Description	462
10.22	mln::accu::meta::histo Struct Reference	462
10.22.1	Detailed Description	463
10.23	mln::accu::meta::label_used Struct Reference	463
10.23.1	Detailed Description	464
10.24	mln::accu::meta::logic::land Struct Reference	464
10.24.1	Detailed Description	465
10.25	mln::accu::meta::logic::land_basic Struct Reference	465
10.25.1	Detailed Description	466
10.26	mln::accu::meta::logic::lor Struct Reference	466
10.26.1	Detailed Description	467
10.27	mln::accu::meta::logic::lor_basic Struct Reference	467
10.27.1	Detailed Description	468
10.28	mln::accu::meta::maj_h Struct Reference	468
10.28.1	Detailed Description	469
10.29	mln::accu::meta::math::count Struct Reference	469

10.29.1 Detailed Description	470
10.30mln::accu::meta::math::inf Struct Reference	470
10.30.1 Detailed Description	471
10.31mln::accu::meta::math::sum Struct Reference	471
10.31.1 Detailed Description	472
10.32mln::accu::meta::math::sup Struct Reference	472
10.32.1 Detailed Description	473
10.33mln::accu::meta::max_site Struct Reference	473
10.33.1 Detailed Description	474
10.34mln::accu::meta::nil Struct Reference	474
10.34.1 Detailed Description	475
10.35mln::accu::meta::p< mA > Struct Template Reference	475
10.35.1 Detailed Description	476
10.36mln::accu::meta::pair< A1, A2 > Struct Template Reference	476
10.36.1 Detailed Description	477
10.37mln::accu::meta::rms Struct Reference	477
10.37.1 Detailed Description	478
10.38mln::accu::meta::shape::bbox Struct Reference	478
10.38.1 Detailed Description	479
10.39mln::accu::meta::shape::height Struct Reference	479
10.39.1 Detailed Description	480
10.40mln::accu::meta::shape::volume Struct Reference	480
10.40.1 Detailed Description	481
10.41mln::accu::meta::stat::max Struct Reference	481
10.41.1 Detailed Description	482
10.42mln::accu::meta::stat::max_h Struct Reference	482
10.42.1 Detailed Description	483
10.43mln::accu::meta::stat::mean Struct Reference	483
10.43.1 Detailed Description	484
10.44mln::accu::meta::stat::median_alt< T > Struct Template Reference	484
10.44.1 Detailed Description	485
10.45mln::accu::meta::stat::median_h Struct Reference	485
10.45.1 Detailed Description	486
10.46mln::accu::meta::stat::min Struct Reference	486
10.46.1 Detailed Description	487
10.47mln::accu::meta::stat::min_h Struct Reference	487

10.47.1 Detailed Description	488
10.48mln::accu::meta::stat::rank Struct Reference	488
10.48.1 Detailed Description	489
10.49mln::accu::meta::stat::rank_high_quant Struct Reference	489
10.49.1 Detailed Description	490
10.50mln::accu::meta::tuple< n, > Struct Template Reference	490
10.50.1 Detailed Description	491
10.51mln::accu::meta::val< mA > Struct Template Reference	491
10.51.1 Detailed Description	492
10.52mln::accu::nil< T > Struct Template Reference	492
10.52.1 Detailed Description	493
10.52.2 Member Function Documentation	493
10.52.2.1 init	493
10.52.2.2 is_valid	493
10.52.2.3 take_as_init	493
10.52.2.4 take_n_times	493
10.52.2.5 to_result	494
10.53mln::accu::p< A > Struct Template Reference	494
10.53.1 Detailed Description	494
10.53.2 Member Function Documentation	494
10.53.2.1 init	494
10.53.2.2 is_valid	495
10.53.2.3 take_as_init	495
10.53.2.4 take_n_times	495
10.53.2.5 to_result	495
10.54mln::accu::pair< A1, A2, T > Struct Template Reference	495
10.54.1 Detailed Description	497
10.54.2 Member Function Documentation	497
10.54.2.1 first	497
10.54.2.2 first_accu	497
10.54.2.3 init	497
10.54.2.4 is_valid	497
10.54.2.5 second	497
10.54.2.6 second_accu	498
10.54.2.7 take_as_init	498
10.54.2.8 take_n_times	498

10.54.2.9 to_result	498
10.55mln::accu::rms< T, V > Struct Template Reference	498
10.55.1 Detailed Description	499
10.55.2 Member Function Documentation	499
10.55.2.1 init	499
10.55.2.2 is_valid	499
10.55.2.3 take_as_init	499
10.55.2.4 take_n_times	499
10.55.2.5 to_result	499
10.56mln::accu::shape::bbox< P > Struct Template Reference	500
10.56.1 Detailed Description	500
10.56.2 Member Function Documentation	500
10.56.2.1 init	500
10.56.2.2 is_valid	500
10.56.2.3 take_as_init	501
10.56.2.4 take_n_times	501
10.56.2.5 to_result	501
10.57mln::accu::shape::height< I > Struct Template Reference	501
10.57.1 Detailed Description	502
10.57.2 Member Typedef Documentation	502
10.57.2.1 argument	502
10.57.2.2 value	502
10.57.3 Member Function Documentation	502
10.57.3.1 init	502
10.57.3.2 is_valid	502
10.57.3.3 set_value	503
10.57.3.4 take_as_init	503
10.57.3.5 take_n_times	503
10.57.3.6 to_result	503
10.58mln::accu::shape::volume< I > Struct Template Reference	503
10.58.1 Detailed Description	504
10.58.2 Member Typedef Documentation	504
10.58.2.1 argument	504
10.58.2.2 value	504
10.58.3 Member Function Documentation	504
10.58.3.1 init	504

10.58.3.2	is_valid	505
10.58.3.3	set_value	505
10.58.3.4	take_as_init	505
10.58.3.5	take_n_times	505
10.58.3.6	to_result	505
10.59	mln::accu::site_set::rectangularity< P > Class Template Reference	505
10.59.1	Detailed Description	506
10.59.2	Constructor & Destructor Documentation	506
10.59.2.1	rectangularity	506
10.59.3	Member Function Documentation	506
10.59.3.1	area	506
10.59.3.2	bbox	506
10.59.3.3	take_as_init	507
10.59.3.4	take_n_times	507
10.59.3.5	to_result	507
10.60	mln::accu::stat::deviation< T, S, M > Struct Template Reference	507
10.60.1	Detailed Description	508
10.60.2	Member Function Documentation	508
10.60.2.1	init	508
10.60.2.2	is_valid	508
10.60.2.3	take_as_init	508
10.60.2.4	take_n_times	508
10.60.2.5	to_result	508
10.61	mln::accu::stat::histo3d_rgb< V > Struct Template Reference	509
10.61.1	Detailed Description	509
10.61.2	Constructor & Destructor Documentation	510
10.61.2.1	histo3d_rgb	510
10.61.3	Member Function Documentation	510
10.61.3.1	init	510
10.61.3.2	is_valid	510
10.61.3.3	take	510
10.61.3.4	take	510
10.61.3.5	take_as_init	511
10.61.3.6	take_n_times	511
10.61.3.7	to_result	511
10.62	mln::accu::stat::max< T > Struct Template Reference	511

10.62.1 Detailed Description	512
10.62.2 Member Function Documentation	512
10.62.2.1 init	512
10.62.2.2 is_valid	512
10.62.2.3 set_value	512
10.62.2.4 take_as_init	512
10.62.2.5 take_n_times	512
10.62.2.6 to_result	513
10.63mln::accu::stat::max_h< V > Struct Template Reference	513
10.63.1 Detailed Description	513
10.63.2 Member Function Documentation	513
10.63.2.1 init	513
10.63.2.2 is_valid	514
10.63.2.3 take_as_init	514
10.63.2.4 take_n_times	514
10.63.2.5 to_result	514
10.64mln::accu::stat::mean< T, S, M > Struct Template Reference	514
10.64.1 Detailed Description	515
10.64.2 Member Function Documentation	515
10.64.2.1 count	515
10.64.2.2 init	515
10.64.2.3 is_valid	515
10.64.2.4 sum	515
10.64.2.5 take_as_init	516
10.64.2.6 take_n_times	516
10.64.2.7 to_result	516
10.65mln::accu::stat::median_alt< S > Struct Template Reference	516
10.65.1 Detailed Description	517
10.65.2 Member Function Documentation	517
10.65.2.1 is_valid	517
10.65.2.2 take	517
10.65.2.3 take_as_init	517
10.65.2.4 take_n_times	517
10.65.2.5 to_result	518
10.66mln::accu::stat::median_h< V > Struct Template Reference	518
10.66.1 Detailed Description	519

10.66.2 Member Function Documentation	519
10.66.2.1 init	519
10.66.2.2 is_valid	519
10.66.2.3 take_as_init	519
10.66.2.4 take_n_times	519
10.66.2.5 to_result	519
10.67mln::accu::stat::meta::deviation Struct Reference	519
10.67.1 Detailed Description	520
10.68mln::accu::stat::min< T > Struct Template Reference	520
10.68.1 Detailed Description	521
10.68.2 Member Function Documentation	521
10.68.2.1 init	521
10.68.2.2 is_valid	521
10.68.2.3 set_value	521
10.68.2.4 take_as_init	521
10.68.2.5 take_n_times	522
10.68.2.6 to_result	522
10.69mln::accu::stat::min_h< V > Struct Template Reference	522
10.69.1 Detailed Description	522
10.69.2 Member Function Documentation	523
10.69.2.1 init	523
10.69.2.2 is_valid	523
10.69.2.3 take_as_init	523
10.69.2.4 take_n_times	523
10.69.2.5 to_result	523
10.70mln::accu::stat::min_max< V > Struct Template Reference	523
10.70.1 Detailed Description	525
10.70.2 Member Function Documentation	525
10.70.2.1 first	525
10.70.2.2 first_accu	525
10.70.2.3 init	525
10.70.2.4 is_valid	525
10.70.2.5 second	525
10.70.2.6 second_accu	525
10.70.2.7 take_as_init	526
10.70.2.8 take_n_times	526

10.70.2.9	to_result	526
10.71	mln::accu::stat::rank< T > Struct Template Reference	526
10.71.1	Detailed Description	527
10.71.2	Member Function Documentation	527
10.71.2.1	init	527
10.71.2.2	is_valid	527
10.71.2.3	k	527
10.71.2.4	take_as_init	527
10.71.2.5	take_n_times	527
10.71.2.6	to_result	528
10.72	mln::accu::stat::rank< bool > Struct Template Reference	528
10.72.1	Detailed Description	528
10.72.2	Member Function Documentation	528
10.72.2.1	init	528
10.72.2.2	is_valid	529
10.72.2.3	take_as_init	529
10.72.2.4	take_n_times	529
10.72.2.5	to_result	529
10.73	mln::accu::stat::rank_high_quant< T > Struct Template Reference	529
10.73.1	Detailed Description	530
10.73.2	Member Function Documentation	530
10.73.2.1	init	530
10.73.2.2	is_valid	530
10.73.2.3	take_as_init	530
10.73.2.4	take_n_times	530
10.73.2.5	to_result	530
10.74	mln::accu::stat::var< T > Struct Template Reference	531
10.74.1	Detailed Description	531
10.74.2	Member Typedef Documentation	532
10.74.2.1	mean_t	532
10.74.3	Member Function Documentation	532
10.74.3.1	init	532
10.74.3.2	is_valid	532
10.74.3.3	mean	532
10.74.3.4	n_items	532
10.74.3.5	take_as_init	532

10.74.3.6	take_n_times	532
10.74.3.7	to_result	533
10.74.3.8	variance	533
10.75	mln::accu::stat::variance< T, S, R > Struct Template Reference	533
10.75.1	Detailed Description	534
10.75.2	Member Function Documentation	534
10.75.2.1	init	534
10.75.2.2	is_valid	534
10.75.2.3	mean	534
10.75.2.4	n_items	534
10.75.2.5	standard_deviation	535
10.75.2.6	sum	535
10.75.2.7	take_as_init	535
10.75.2.8	take_n_times	535
10.75.2.9	to_result	535
10.75.2.10	var	535
10.76	mln::accu::tuple< A, n, > Struct Template Reference	535
10.76.1	Detailed Description	536
10.76.2	Member Function Documentation	536
10.76.2.1	init	536
10.76.2.2	is_valid	536
10.76.2.3	take_as_init	537
10.76.2.4	take_n_times	537
10.76.2.5	to_result	537
10.77	mln::accu::val< A > Struct Template Reference	537
10.77.1	Detailed Description	538
10.77.2	Member Function Documentation	538
10.77.2.1	init	538
10.77.2.2	is_valid	538
10.77.2.3	take_as_init	538
10.77.2.4	take_n_times	538
10.77.2.5	to_result	538
10.78	mln::Accumulator< E > Struct Template Reference	538
10.78.1	Detailed Description	540
10.78.2	Member Function Documentation	540
10.78.2.1	take_as_init	540

10.78.2.2 take_n_times	540
10.79mln::algebra::h_mat< d, T > Struct Template Reference	540
10.79.1 Detailed Description	541
10.79.2 Member Enumeration Documentation	541
10.79.2.1 "@7	541
10.79.3 Constructor & Destructor Documentation	541
10.79.3.1 h_mat	541
10.79.3.2 h_mat	541
10.79.4 Member Function Documentation	542
10.79.4.1 _1	542
10.79.4.2 t	542
10.80mln::algebra::h_vec< d, C > Class Template Reference	542
10.80.1 Detailed Description	543
10.80.2 Member Enumeration Documentation	543
10.80.2.1 "@8	543
10.80.3 Constructor & Destructor Documentation	543
10.80.3.1 h_vec	543
10.80.3.2 h_vec	543
10.80.4 Member Function Documentation	543
10.80.4.1 operator mat< n, 1, U >	543
10.80.4.2 t	544
10.80.4.3 to_vec	544
10.80.5 Member Data Documentation	544
10.80.5.1 origin	544
10.80.5.2 zero	544
10.81mln::bkd_pixter1d< I > Class Template Reference	544
10.81.1 Detailed Description	545
10.81.2 Member Typedef Documentation	545
10.81.2.1 image	545
10.81.3 Constructor & Destructor Documentation	545
10.81.3.1 bkd_pixter1d	545
10.81.4 Member Function Documentation	545
10.81.4.1 next	545
10.82mln::bkd_pixter2d< I > Class Template Reference	545
10.82.1 Detailed Description	546
10.82.2 Member Typedef Documentation	546

10.82.2.1 image	546
10.82.3 Constructor & Destructor Documentation	546
10.82.3.1 bkd_pixter2d	546
10.82.4 Member Function Documentation	546
10.82.4.1 next	546
10.83mln::bkd_pixter3d< I > Class Template Reference	547
10.83.1 Detailed Description	547
10.83.2 Member Typedef Documentation	547
10.83.2.1 image	547
10.83.3 Constructor & Destructor Documentation	548
10.83.3.1 bkd_pixter3d	548
10.83.4 Member Function Documentation	548
10.83.4.1 next	548
10.84mln::box< P > Class Template Reference	548
10.84.1 Detailed Description	551
10.84.2 Member Typedef Documentation	551
10.84.2.1 bkd_piter	551
10.84.2.2 element	551
10.84.2.3 fwd_piter	552
10.84.2.4 piter	552
10.84.2.5 psite	552
10.84.2.6 site	552
10.84.3 Member Enumeration Documentation	552
10.84.3.1 "@31	552
10.84.4 Constructor & Destructor Documentation	552
10.84.4.1 box	552
10.84.4.2 box	552
10.84.4.3 box	552
10.84.5 Member Function Documentation	553
10.84.5.1 bbox	553
10.84.5.2 crop_wrt	553
10.84.5.3 enlarge	553
10.84.5.4 enlarge	553
10.84.5.5 has	553
10.84.5.6 is_empty	554
10.84.5.7 is_valid	554

10.84.5.8 len	554
10.84.5.9 memory_size	554
10.84.5.10 merge	554
10.84.5.11 nsites	554
10.84.5.12 pcenter	555
10.84.5.13 pmax	555
10.84.5.14 pmax	555
10.84.5.15 pmin	555
10.84.5.16 pmin	555
10.84.5.17 to_larger	555
10.84.6 Friends And Related Function Documentation	556
10.84.6.1 operator<<	556
10.85 mln::Box< E > Struct Template Reference	556
10.85.1 Detailed Description	558
10.85.2 Member Function Documentation	559
10.85.2.1 bbox	559
10.85.2.2 is_empty	559
10.85.2.3 len	559
10.85.2.4 nsites	559
10.85.3 Friends And Related Function Documentation	560
10.85.3.1 diff	560
10.85.3.2 inter	560
10.85.3.3 operator<	560
10.85.3.4 operator<	560
10.85.3.5 operator<<	560
10.85.3.6 operator<=	561
10.85.3.7 operator<=	561
10.85.3.8 operator==	561
10.85.3.9 sym_diff	561
10.85.3.10 uni	561
10.85.3.11 unique	562
10.86 mln::box_runend_piter< P > Class Template Reference	562
10.86.1 Detailed Description	562
10.86.2 Member Function Documentation	562
10.86.2.1 next	562
10.86.2.2 run_length	562

10.87mln::box_runstart_piter< P > Class Template Reference	563
10.87.1 Detailed Description	563
10.87.2 Constructor & Destructor Documentation	563
10.87.2.1 box_runstart_piter	563
10.87.3 Member Function Documentation	563
10.87.3.1 next	563
10.87.3.2 run_length	564
10.88mln::Browsing< E > Struct Template Reference	564
10.88.1 Detailed Description	564
10.89mln::canvas::browsing::backdiagonal2d_t Struct Reference	565
10.89.1 Detailed Description	565
10.90mln::canvas::browsing::breadth_first_search_t Struct Reference	566
10.90.1 Detailed Description	566
10.91mln::canvas::browsing::depth_first_search_t Struct Reference	566
10.91.1 Detailed Description	566
10.92mln::canvas::browsing::diagonal2d_t Struct Reference	567
10.92.1 Detailed Description	567
10.93mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference	568
10.93.1 Detailed Description	569
10.94mln::canvas::browsing::directional_t Struct Reference	569
10.94.1 Detailed Description	570
10.95mln::canvas::browsing::fwd_t Struct Reference	571
10.95.1 Detailed Description	572
10.96mln::canvas::browsing::hyper_directional_t Struct Reference	572
10.96.1 Detailed Description	573
10.97mln::canvas::browsing::snake_fwd_t Struct Reference	574
10.97.1 Detailed Description	574
10.98mln::canvas::browsing::snake_generic_t Struct Reference	575
10.98.1 Detailed Description	576
10.99mln::canvas::browsing::snake_vert_t Struct Reference	577
10.99.1 Detailed Description	577
10.100mln::canvas::chamfer< F > Struct Template Reference	578
10.100.1 Detailed Description	578
10.101mln::category< R(*) (A) > Struct Template Reference	578
10.101.1 Detailed Description	578
10.102mln::complex_image< D, G, V > Class Template Reference	579

10.102.Detailed Description	580
10.102.2Member Typedef Documentation	580
10.102.2.1geom	580
10.102.2.2value	580
10.102.2.3value	580
10.102.2.4skeleton	580
10.102.2.5value	580
10.102.3Constructor & Destructor Documentation	581
10.102.3.1complex_image	581
10.102.4Member Function Documentation	581
10.102.4.1domain	581
10.102.4.2operator()	581
10.102.4.3operator()	581
10.102.4.4values	581
10.102.5Member Data Documentation	581
10.102.5.1dim	581
10.103In::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference	582
10.103.Detailed Description	582
10.103.2Member Typedef Documentation	582
10.103.2.1iter_type	582
10.103.2.2psite	583
10.103.3Constructor & Destructor Documentation	583
10.103.3.1complex_neighborhood_bkd_piter	583
10.103.4Member Function Documentation	583
10.103.4.1iter	583
10.103.4.2next	583
10.104In::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference	583
10.104.Detailed Description	584
10.104.2Member Typedef Documentation	584
10.104.2.1iter_type	584
10.104.2.2psite	584
10.104.3Constructor & Destructor Documentation	584
10.104.3.1complex_neighborhood_fwd_piter	584
10.104.4Member Function Documentation	585
10.104.4.1iter	585
10.104.4.2next	585

10.105.1. <code>ln::complex_psite< D, G ></code> Class Template Reference	585
10.105.2. Detailed Description	586
10.105.3. Constructor & Destructor Documentation	586
10.105.4. <code>lcomplex_psite</code>	586
10.105.5. <code>2complex_psite</code>	586
10.105.6. Member Function Documentation	586
10.105.7. <code>lchange_target</code>	586
10.105.8. <code>2face</code>	587
10.105.9. <code>3face_id</code>	587
10.105.10. <code>4invalidate</code>	587
10.105.11. <code>5is_valid</code>	587
10.105.12. <code>6</code>	587
10.105.13. <code>7site_set</code>	588
10.106.1. <code>ln::complex_window_bkd_piter< I, G, W ></code> Class Template Reference	588
10.106.2. Detailed Description	588
10.106.3. Member Typedef Documentation	589
10.106.4. <code>liter_type</code>	589
10.106.5. <code>2psite</code>	589
10.106.6. Constructor & Destructor Documentation	589
10.106.7. <code>lcomplex_window_bkd_piter</code>	589
10.106.8. Member Function Documentation	589
10.106.9. <code>liter</code>	589
10.106.10. <code>2next</code>	589
10.107.1. <code>ln::complex_window_fwd_piter< I, G, W ></code> Class Template Reference	590
10.107.2. Detailed Description	590
10.107.3. Member Typedef Documentation	590
10.107.4. <code>liter_type</code>	590
10.107.5. <code>2psite</code>	591
10.107.6. Constructor & Destructor Documentation	591
10.107.7. <code>lcomplex_window_fwd_piter</code>	591
10.107.8. Member Function Documentation	591
10.107.9. <code>liter</code>	591
10.107.10. <code>2next</code>	591
10.108.1. <code>ln::decorated_image< I, D ></code> Struct Template Reference	591
10.108.2. Detailed Description	592
10.108.3. Member Typedef Documentation	592

10.108.2.1lvalue	592
10.108.2.2psite	593
10.108.2.3rvalue	593
10.108.2.4skeleton	593
10.108.3.Constructor & Destructor Documentation	593
10.108.3.1decorated_image	593
10.108.4.Member Function Documentation	593
10.108.4.1decoration	593
10.108.4.2decoration	593
10.108.4.3operator decorated_image< const I, D >	593
10.108.4.4operator()	594
10.108.4.5operator()	594
10.109.Inn::Delta_Point_Site< E > Struct Template Reference	594
10.109.Detailed Description	594
10.110.Inn::Delta_Point_Site< void > Struct Template Reference	595
10.110.Detailed Description	595
10.111.Inn::doc::Accumulator< E > Struct Template Reference	595
10.111.Detailed Description	595
10.111.2.Member Typedef Documentation	596
10.111.2.1argument	596
10.111.3.Member Function Documentation	596
10.111.3.1init	596
10.111.3.2ake	596
10.111.3.3ake	596
10.112.Inn::doc::Box< E > Struct Template Reference	596
10.112.Detailed Description	597
10.112.2.Member Typedef Documentation	597
10.112.2.1bkd_piter	597
10.112.2.2fwd_piter	598
10.112.2.3psite	598
10.112.2.4site	598
10.112.3.Member Function Documentation	598
10.112.3.1bbox	598
10.112.3.2has	598
10.112.3.3nsites	598
10.112.3.4pmax	599

10.112.3.5pmin	599
10.113.1.1nln::doc::Dpoint< E > Struct Template Reference	599
10.113.2.1Detailed Description	600
10.113.3.1Member Typedef Documentation	600
10.113.3.2.1coord	600
10.113.3.2.2dpoint	600
10.113.3.2.3point	600
10.113.3.3Member Enumeration Documentation	600
10.113.3.3.1"@19	600
10.113.3.4Member Function Documentation	601
10.113.3.4.1operator[]	601
10.114.1.1nln::doc::Fastest_Image< E > Struct Template Reference	601
10.114.2.1Detailed Description	603
10.114.3.1Member Typedef Documentation	604
10.114.3.2.1bkd_piter	604
10.114.3.2.2coord	604
10.114.3.2.3dpoint	604
10.114.3.2.4fwd_piter	604
10.114.3.2.5value	604
10.114.3.2.6point	604
10.114.3.2.7pset	605
10.114.3.2.8psite	605
10.114.3.2.9value	605
10.114.3.2.10keleton	605
10.114.3.2.11value	605
10.114.3.2.12set	605
10.114.3.3Member Function Documentation	606
10.114.3.3.1bbox	606
10.114.3.3.2border	606
10.114.3.3.3buffer	606
10.114.3.3.4delta_index	606
10.114.3.3.5domain	606
10.114.3.3.6has	607
10.114.3.3.7has	607
10.114.3.3.8s_valid	607
10.114.3.3.9elements	607

10.114.3.10sites	607
10.114.3.11operator()	607
10.114.3.12operator()	608
10.114.3.13operator[]	608
10.114.3.14operator[]	608
10.114.3.15point_at_index	609
10.114.3.16values	609
10.115.mln::doc::Generalized_Pixel< E > Struct Template Reference	609
10.115.Detailed Description	610
10.115.Member Typedef Documentation	610
10.115.2.limage	610
10.115.2.2value	610
10.115.2.3value	610
10.115.3.Member Function Documentation	611
10.115.3.1ima	611
10.115.3.2val	611
10.116.mln::doc::Image< E > Struct Template Reference	611
10.116.Detailed Description	613
10.116.Member Typedef Documentation	613
10.116.2.lbkd_piter	613
10.116.2.2coord	613
10.116.2.3dpoint	613
10.116.2.4fwd_piter	614
10.116.2.5value	614
10.116.2.6point	614
10.116.2.7pset	614
10.116.2.8psite	614
10.116.2.9value	614
10.116.2.10keleton	615
10.116.2.11value	615
10.116.2.12set	615
10.116.3.Member Function Documentation	615
10.116.3.1bbox	615
10.116.3.2domain	615
10.116.3.3has	615
10.116.3.4has	616

10.116.3.5s_valid	616
10.116.3.6nsites	616
10.116.3.7operator()	616
10.116.3.8operator()	616
10.116.3.9values	617
10.117ln::doc::Iterator< E > Struct Template Reference	617
10.117.Detailed Description	617
10.117.Member Function Documentation	618
10.117.2.invalidate	618
10.117.2.2s_valid	618
10.117.2.3start	618
10.118ln::doc::Neighborhood< E > Struct Template Reference	618
10.118.Detailed Description	619
10.118.Member Typedef Documentation	619
10.118.2.lbkdniter	619
10.118.2.2dpoint	619
10.118.2.3fwd_niter	619
10.118.2.4niter	619
10.118.2.5point	619
10.119ln::doc::Object< E > Struct Template Reference	620
10.119.Detailed Description	620
10.120ln::doc::Pixel_Iterator< E > Struct Template Reference	620
10.120.Detailed Description	622
10.120.Member Typedef Documentation	622
10.120.2.limage	622
10.120.2.2value	622
10.120.2.3rvalue	622
10.120.2.4value	622
10.120.3.Member Function Documentation	622
10.120.3.lima	622
10.120.3.2invalidate	623
10.120.3.3s_valid	623
10.120.3.4start	623
10.120.3.5val	623
10.121ln::doc::Point_Site< E > Struct Template Reference	623
10.121.Detailed Description	624

10.121.2Member Typedef Documentation	624
10.121.2.1coord	624
10.121.2.2dpoint	624
10.121.2.3mesh	624
10.121.2.4point	625
10.121.3Member Enumeration Documentation	625
10.121.3.1"@20	625
10.121.4Member Function Documentation	625
10.121.4.1operator[]	625
10.121.4.2to_point	625
10.122In::doc::Site_Iterator< E > Struct Template Reference	626
10.122.Detailed Description	627
10.122.2Member Typedef Documentation	627
10.122.2.1psite	627
10.122.3Member Function Documentation	627
10.122.3.1invalidate	627
10.122.3.2is_valid	627
10.122.3.3operator psite	627
10.122.3.4start	627
10.123In::doc::Site_Set< E > Struct Template Reference	628
10.123.Detailed Description	629
10.123.2Member Typedef Documentation	629
10.123.2.1bkd_piter	629
10.123.2.2fwd_piter	629
10.123.2.3psite	629
10.123.2.4site	629
10.123.3Member Function Documentation	629
10.123.3.1has	629
10.124In::doc::Value_Iterator< E > Struct Template Reference	630
10.124.Detailed Description	631
10.124.2Member Typedef Documentation	631
10.124.2.1value	631
10.124.3Member Function Documentation	631
10.124.3.1invalidate	631
10.124.3.2is_valid	631
10.124.3.3operator value	631

10.124.3.4start	631
10.125. Struct Template Reference	631
10.125. Detailed Description	632
10.125. Member Typedef Documentation	633
10.125.2.lbkd_viter	633
10.125.2.fwd_viter	633
10.125.2.value	633
10.125. Member Function Documentation	633
10.125.3.lhas	633
10.125.3.index_of	633
10.125.3.nvalues	633
10.125.3.operator[]	634
10.126. Struct Template Reference	634
10.126. Detailed Description	635
10.126. Member Typedef Documentation	635
10.126.2.lbkd_qiter	635
10.126.2.dpoint	635
10.126.2.fwd_qiter	635
10.126.2.point	636
10.126.2.weight	636
10.126.2.window	636
10.126. Member Function Documentation	636
10.126.3.idelta	636
10.126.3.is_centered	636
10.126.3.is_empty	636
10.126.3.sym	636
10.126.3.win	636
10.127. Struct Template Reference	636
10.127. Detailed Description	637
10.127. Member Typedef Documentation	637
10.127.2.lbkd_qiter	637
10.127.2.fwd_qiter	638
10.127.2.qiter	638
10.128. Dpoint< E > Struct Template Reference	638
10.128. Detailed Description	638
10.128. Member Function Documentation	639

10.128.2. <code>lto_dpoint</code>	639
10.129. <code>mln::dpoint< G, C ></code> Struct Template Reference	639
10.129. Detailed Description	641
10.129. Member Typedef Documentation	641
10.129.2. <code>lcoord</code>	641
10.129.2. <code>2grid</code>	641
10.129.2. <code>3psite</code>	641
10.129.2. <code>4site</code>	641
10.129.2. <code>5vec</code>	641
10.129.3. Member Enumeration Documentation	642
10.129.3. <code>l"@22</code>	642
10.129.4. Constructor & Destructor Documentation	642
10.129.4. <code>ldpoint</code>	642
10.129.4. <code>2dpoint</code>	642
10.129.4. <code>3dpoint</code>	642
10.129.4. <code>4dpoint</code>	642
10.129.4. <code>5dpoint</code>	642
10.129.5. Member Function Documentation	643
10.129.5. <code>operator mln::algebra::vec< dpoint< G, C >::dim, Q ></code>	643
10.129.5. <code>operator[]</code>	643
10.129.5. <code>operator[]</code>	643
10.129.5. <code>set_all</code>	643
10.129.5. <code>to_vec</code>	644
10.130. <code>mln::dpoints_bkd_pixter< I ></code> Class Template Reference	644
10.130. Detailed Description	645
10.130. Constructor & Destructor Documentation	645
10.130.2. <code>ldpoints_bkd_pixter</code>	645
10.130.2. <code>2dpoints_bkd_pixter</code>	645
10.130.3. Member Function Documentation	645
10.130.3. <code>lcenter_val</code>	645
10.130.3. <code>2invalidate</code>	645
10.130.3. <code>3is_valid</code>	646
10.130.3. <code>4next</code>	646
10.130.3. <code>5start</code>	646
10.130.3. <code>6update</code>	646
10.131. <code>mln::dpoints_fwd_pixter< I ></code> Class Template Reference	646

10.131.Detailed Description	647
10.131.Constructor & Destructor Documentation	647
10.131.2.1dpoints_fwd_piter	647
10.131.2.2dpoints_fwd_piter	648
10.131.3.Member Function Documentation	648
10.131.3.1center_val	648
10.131.3.2invalidate	648
10.131.3.3is_valid	648
10.131.3.4next	648
10.131.3.5start	649
10.131.3.6update	649
10.132.mln::dpsites_bkd_piter< V > Class Template Reference	649
10.132.Detailed Description	649
10.132.Constructor & Destructor Documentation	650
10.132.2.1dpsites_bkd_piter	650
10.132.2.2dpsites_bkd_piter	650
10.132.3.Member Function Documentation	650
10.132.3.1next	650
10.133.mln::dpsites_fwd_piter< V > Class Template Reference	650
10.133.Detailed Description	651
10.133.Constructor & Destructor Documentation	651
10.133.2.1dpsites_fwd_piter	651
10.133.2.2dpsites_fwd_piter	651
10.133.3.Member Function Documentation	651
10.133.3.1next	651
10.134.mln::Edge< E > Struct Template Reference	652
10.134.Detailed Description	652
10.135.mln::edge_image< P, V, G > Class Template Reference	652
10.135.Detailed Description	653
10.135.2.Member Typedef Documentation	653
10.135.2.1edge_nbh_t	653
10.135.2.2edge_win_t	653
10.135.2.3graph_t	653
10.135.2.4nbh_t	653
10.135.2.5site_function_t	654
10.135.2.6skeleton	654

10.135.2.7win_t	654
10.135.3.Constructor & Destructor Documentation	654
10.135.3.1edge_image	654
10.135.4.Member Function Documentation	654
10.135.4.1operator()	654
10.136.mln::extended< I > Struct Template Reference	654
10.136.1.Detailed Description	655
10.136.2.Member Typedef Documentation	655
10.136.2.1skeleton	655
10.136.2.2value	655
10.136.3.Constructor & Destructor Documentation	655
10.136.3.1extended	655
10.136.3.2extended	656
10.136.4.Member Function Documentation	656
10.136.4.1domain	656
10.137.mln::extension_fun< I, F > Class Template Reference	656
10.137.1.Detailed Description	657
10.137.2.Member Typedef Documentation	657
10.137.2.1rvalue	657
10.137.2.2skeleton	657
10.137.2.3value	657
10.137.3.Constructor & Destructor Documentation	657
10.137.3.1extension_fun	657
10.137.3.2extension_fun	658
10.137.4.Member Function Documentation	658
10.137.4.1extension	658
10.137.4.2has	658
10.137.4.3operator()	658
10.137.4.4operator()	658
10.138.mln::extension_ima< I, J > Class Template Reference	658
10.138.1.Detailed Description	659
10.138.2.Member Typedef Documentation	659
10.138.2.1rvalue	659
10.138.2.2skeleton	660
10.138.2.3value	660
10.138.3.Constructor & Destructor Documentation	660

10.138.3.1extension_ima	660
10.138.3.2extension_ima	660
10.138.4Member Function Documentation	660
10.138.4.1extension	660
10.138.4.2has	660
10.138.4.3operator()	660
10.138.4.4operator()	661
10.139Inn::extension_val< I > Class Template Reference	661
10.139.1Detailed Description	662
10.139.2Member Typedef Documentation	662
10.139.2.1rvalue	662
10.139.2.2skeleton	662
10.139.2.3value	662
10.139.3Constructor & Destructor Documentation	662
10.139.3.1extension_val	662
10.139.3.2extension_val	662
10.139.4Member Function Documentation	662
10.139.4.1change_extension	662
10.139.4.2extension	663
10.139.4.3has	663
10.139.4.4operator()	663
10.139.4.5operator()	663
10.140Inn::flat_image< T, S > Struct Template Reference	663
10.140.1Detailed Description	664
10.140.2Member Typedef Documentation	664
10.140.2.1lvalue	664
10.140.2.2rvalue	664
10.140.2.3skeleton	664
10.140.2.4value	665
10.140.3Constructor & Destructor Documentation	665
10.140.3.1flat_image	665
10.140.3.2flat_image	665
10.140.4Member Function Documentation	665
10.140.4.1domain	665
10.140.4.2has	665
10.140.4.3operator()	665

10.140.4.4.operator()	665
10.141.mln::fun::from_accu< A > Struct Template Reference	666
10.141.Detailed Description	666
10.142.mln::fun::n2v::white_gaussian< V > Struct Template Reference	666
10.142.Detailed Description	667
10.143.mln::fun::p2b::antilogy Struct Reference	667
10.143.Detailed Description	668
10.144.mln::fun::p2b::tautology Struct Reference	668
10.144.Detailed Description	669
10.145.mln::fun::v2b::lnot< V > Struct Template Reference	670
10.145.Detailed Description	670
10.146.mln::fun::v2b::threshold< V > Struct Template Reference	671
10.146.Detailed Description	671
10.147.mln::fun::v2v::ch_function_value< F, V > Class Template Reference	672
10.147.Detailed Description	672
10.148.mln::fun::v2v::component< T, i > Struct Template Reference	672
10.148.Detailed Description	673
10.149.mln::fun::v2v::l1_norm< V, R > Struct Template Reference	673
10.149.Detailed Description	674
10.150.mln::fun::v2v::l2_norm< V, R > Struct Template Reference	674
10.150.Detailed Description	675
10.151.mln::fun::v2v::linear< V, T, R > Struct Template Reference	675
10.151.Detailed Description	676
10.152.mln::fun::v2v::linfty_norm< V, R > Struct Template Reference	676
10.152.Detailed Description	677
10.153.mln::fun::v2v::rgb8_to_rgbn< n > Struct Template Reference	677
10.153.Detailed Description	678
10.153.Member Function Documentation	679
10.153.2.operator()	679
10.154.mln::fun::v2w2v::cos< V > Struct Template Reference	679
10.154.Detailed Description	680
10.155.mln::fun::v2w_w2v::l1_norm< V, R > Struct Template Reference	680
10.155.Detailed Description	681
10.156.mln::fun::v2w_w2v::l2_norm< V, R > Struct Template Reference	682
10.156.Detailed Description	682
10.157.mln::fun::v2w_w2v::linfty_norm< V, R > Struct Template Reference	683

10.157. Detailed Description	683
10.158. <code>ln::fun::vv2b::eq< L, R > Struct Template Reference</code>	684
10.158. Detailed Description	684
10.159. <code>ln::fun::vv2b::ge< L, R > Struct Template Reference</code>	684
10.159. Detailed Description	685
10.160. <code>ln::fun::vv2b::gt< L, R > Struct Template Reference</code>	685
10.160. Detailed Description	686
10.161. <code>ln::fun::vv2b::implies< L, R > Struct Template Reference</code>	686
10.161. Detailed Description	687
10.162. <code>ln::fun::vv2b::le< L, R > Struct Template Reference</code>	687
10.162. Detailed Description	688
10.163. <code>ln::fun::vv2b::lt< L, R > Struct Template Reference</code>	688
10.163. Detailed Description	689
10.164. <code>ln::fun::vv2v::diff_abs< V > Struct Template Reference</code>	689
10.164. Detailed Description	690
10.165. <code>ln::fun::vv2v::land< L, R > Struct Template Reference</code>	690
10.165. Detailed Description	691
10.166. <code>ln::fun::vv2v::land_not< L, R > Struct Template Reference</code>	691
10.166. Detailed Description	692
10.167. <code>ln::fun::vv2v::lor< L, R > Struct Template Reference</code>	692
10.167. Detailed Description	693
10.168. <code>ln::fun::vv2v::lxor< L, R > Struct Template Reference</code>	693
10.168. Detailed Description	694
10.169. <code>ln::fun::vv2v::max< V > Struct Template Reference</code>	694
10.169. Detailed Description	695
10.170. <code>ln::fun::vv2v::min< L, R > Struct Template Reference</code>	695
10.170. Detailed Description	696
10.171. <code>ln::fun::vv2v::vec< V > Struct Template Reference</code>	696
10.171. Detailed Description	697
10.172. <code>ln::fun::x2p::closest_point< P > Struct Template Reference</code>	697
10.172. Detailed Description	698
10.173. <code>ln::fun::x2v::bilinear< I > Struct Template Reference</code>	698
10.173. Detailed Description	698
10.173.2. Member Function Documentation	698
10.173.2.1. <code>operator()</code>	698
10.173.2.2. <code>operator()</code>	699

10.174	fun::x2v::trilinear< I > Struct Template Reference	699
10.174.	Detailed Description	699
10.175	fun::x2x::composed< T2, T1 > Struct Template Reference	699
10.175.	Detailed Description	699
10.175.	Constructor & Destructor Documentation	700
10.175.2.	lcomposed	700
10.175.2.	xcomposed	700
10.176	fun::x2x::linear< I > Struct Template Reference	700
10.176.	Detailed Description	701
10.176.	Constructor & Destructor Documentation	701
10.176.2.	llinear	701
10.176.	Member Function Documentation	701
10.176.3.	operator()	701
10.176.	Member Data Documentation	701
10.176.4.	lima	701
10.177	fun::x2x::rotation< n, C > Struct Template Reference	701
10.177.	Detailed Description	703
10.177.	Member Typedef Documentation	703
10.177.2.	ldata_t	703
10.177.2.	invert	703
10.177.	Constructor & Destructor Documentation	703
10.177.3.	lrotation	703
10.177.3.	rotation	704
10.177.3.	rotation	704
10.177.3.	rotation	704
10.177.	Member Function Documentation	704
10.177.4.	linv	704
10.177.4.	operator()	704
10.177.4.	set_alpha	704
10.177.4.	set_axis	704
10.178	fun::x2x::translation< n, C > Struct Template Reference	705
10.178.	Detailed Description	706
10.178.	Member Typedef Documentation	706
10.178.2.	ldata_t	706
10.178.2.	invert	706
10.178.	Constructor & Destructor Documentation	706

10.178.3.1translation	706
10.178.3.2translation	707
10.178.4Member Function Documentation	707
10.178.4.1inv	707
10.178.4.2operator()	707
10.178.4.3set_t	707
10.178.4.4t	707
10.179Inn::fun_image< F, I > Struct Template Reference	707
10.179.Detailed Description	708
10.179.2Member Typedef Documentation	708
10.179.2.1lvalue	708
10.179.2.2rvalue	708
10.179.2.3skeleton	709
10.179.2.4value	709
10.179.3Constructor & Destructor Documentation	709
10.179.3.1fun_image	709
10.179.3.2fun_image	709
10.179.3.3fun_image	709
10.179.4Member Function Documentation	709
10.179.4.1operator()	709
10.179.4.2operator()	709
10.180Inn::Function< E > Struct Template Reference	710
10.180.Detailed Description	710
10.180.2Constructor & Destructor Documentation	710
10.180.2.1Function	710
10.181Inn::Function< void > Struct Template Reference	710
10.181.Detailed Description	710
10.182Inn::Function_n2v< E > Struct Template Reference	711
10.182.Detailed Description	711
10.183Inn::Function_v2b< E > Struct Template Reference	711
10.183.Detailed Description	712
10.184Inn::Function_v2v< E > Struct Template Reference	712
10.184.Detailed Description	713
10.185Inn::Function_vv2b< E > Struct Template Reference	713
10.185.Detailed Description	713
10.186Inn::Function_vv2v< E > Struct Template Reference	713

10.186.Detailed Description	714
10.187ln::fwd_pixter1d< I > Class Template Reference	714
10.187.Detailed Description	715
10.187.Member Typedef Documentation	715
10.187.2.limage	715
10.187.Constructor & Destructor Documentation	715
10.187.3.lfwd_pixter1d	715
10.187.Member Function Documentation	715
10.187.4.lnext	715
10.188ln::fwd_pixter2d< I > Class Template Reference	715
10.188.Detailed Description	716
10.188.Member Typedef Documentation	716
10.188.2.limage	716
10.188.Constructor & Destructor Documentation	716
10.188.3.lfwd_pixter2d	716
10.188.Member Function Documentation	717
10.188.4.lnext	717
10.189ln::fwd_pixter3d< I > Class Template Reference	717
10.189.Detailed Description	717
10.189.Member Typedef Documentation	717
10.189.2.limage	717
10.189.Constructor & Destructor Documentation	718
10.189.3.lfwd_pixter3d	718
10.189.Member Function Documentation	718
10.189.4.lnext	718
10.190ln::Gdpoint< E > Struct Template Reference	718
10.190.Detailed Description	719
10.191ln::Gdpoint< void > Struct Template Reference	719
10.191.Detailed Description	719
10.192ln::Generalized_Pixel< E > Struct Template Reference	720
10.192.Detailed Description	720
10.193ln::geom::complex_geometry< D, P > Class Template Reference	720
10.193.Detailed Description	721
10.193.Constructor & Destructor Documentation	721
10.193.2.lcomplex_geometry	721
10.193.Member Function Documentation	721

10.193.3.ladd_location	721
10.193.3.2operator()	722
10.194.mln::Gpoint< E > Struct Template Reference	722
10.194.Detailed Description	723
10.194.Friends And Related Function Documentation	723
10.194.2.1operator+	723
10.194.2.2operator+=	724
10.194.2.3operator-	724
10.194.2.4operator-=	725
10.194.2.5operator/	725
10.194.2.6operator<<	725
10.194.2.7operator==	726
10.195.mln::Graph< E > Struct Template Reference	726
10.195.Detailed Description	726
10.196.mln::graph::attribute::card_t Struct Reference	727
10.196.Detailed Description	727
10.196.Member Typedef Documentation	727
10.196.2.lresult	727
10.197.mln::graph::attribute::representative_t Struct Reference	727
10.197.Detailed Description	728
10.197.Member Typedef Documentation	728
10.197.2.lresult	728
10.198.mln::graph_elt_mixed_neighborhood< G, S, S2 > Struct Template Reference	728
10.198.Detailed Description	729
10.198.Member Typedef Documentation	729
10.198.2.1bkd_niter	729
10.198.2.2fwd_niter	729
10.198.2.3niter	729
10.199.mln::graph_elt_mixed_window< G, S, S2 > Class Template Reference	729
10.199.Detailed Description	731
10.199.Member Typedef Documentation	731
10.199.2.1bkd_qiter	731
10.199.2.2center_t	731
10.199.2.3fwd_qiter	732
10.199.2.4graph_element	732
10.199.2.5psite	732

10.199.2.6qiter	732
10.199.2.7site	732
10.199.2.8target	732
10.199.3Member Function Documentation	732
10.199.3.1delta	732
10.199.3.2is_centered	733
10.199.3.3is_empty	733
10.199.3.4is_symmetric	733
10.199.3.5is_valid	733
10.199.3.6sym	733
10.200In::graph_elt_neighborhood< G, S > Struct Template Reference	733
10.200.Detailed Description	734
10.200.2Member Typedef Documentation	735
10.200.2.1bkd_niter	735
10.200.2.2fwd_niter	735
10.200.2.3niter	735
10.201In::graph_elt_neighborhood_if< G, S, I > Struct Template Reference	735
10.201.Detailed Description	736
10.201.2Member Typedef Documentation	736
10.201.2.1bkd_niter	736
10.201.2.2fwd_niter	736
10.201.2.3niter	737
10.201.3Constructor & Destructor Documentation	737
10.201.3.1graph_elt_neighborhood_if	737
10.201.3.2graph_elt_neighborhood_if	737
10.201.4Member Function Documentation	737
10.201.4.1mask	737
10.202In::graph_elt_window< G, S > Class Template Reference	737
10.202.Detailed Description	739
10.202.2Member Typedef Documentation	739
10.202.2.1bkd_qiter	739
10.202.2.2center_t	739
10.202.2.3fwd_qiter	740
10.202.2.4graph_element	740
10.202.2.5psite	740
10.202.2.6qiter	740

10.202.2.7site	740
10.202.2.8target	740
10.202.3Member Function Documentation	740
10.202.3.1delta	740
10.202.3.2is_centered	741
10.202.3.3is_empty	741
10.202.3.4is_symmetric	741
10.202.3.5is_valid	741
10.202.3.6sym	741
10.203.1Inn::graph_elt_window_if< G, S, I > Class Template Reference	741
10.203.1Detailed Description	743
10.203.2Member Typedef Documentation	743
10.203.2.1bkd_qiter	743
10.203.2.2fwd_qiter	744
10.203.2.3mask_t	744
10.203.2.4psite	744
10.203.2.5qiter	744
10.203.2.6site	744
10.203.2.7target	744
10.203.3Constructor & Destructor Documentation	745
10.203.3.1graph_elt_window_if	745
10.203.3.2graph_elt_window_if	745
10.203.4Member Function Documentation	745
10.203.4.1change_mask	745
10.203.4.2delta	745
10.203.4.3is_centered	745
10.203.4.4is_empty	745
10.203.4.5is_symmetric	746
10.203.4.6is_valid	746
10.203.4.7mask	746
10.203.4.8sym	746
10.204.1Inn::graph_window_base< P, E > Class Template Reference	746
10.204.1Detailed Description	747
10.204.2Member Typedef Documentation	747
10.204.2.1site	747
10.204.3Member Function Documentation	747

10.204.3.1delta	747
10.204.3.2is_centered	747
10.204.3.3is_empty	748
10.204.3.4is_symmetric	748
10.204.3.5is_valid	748
10.204.3.6sym	748
10.205ln::graph_window_if_piter< S, W, I > Class Template Reference	748
10.205.1Detailed Description	749
10.205.2Member Typedef Documentation	749
10.205.2.1P	749
10.205.3Constructor & Destructor Documentation	749
10.205.3.1graph_window_if_piter	749
10.205.4Member Function Documentation	749
10.205.4.1element	749
10.205.4.2id	749
10.205.4.3next	750
10.206ln::graph_window_piter< S, W, I > Class Template Reference	750
10.206.1Detailed Description	751
10.206.2Member Typedef Documentation	751
10.206.2.1center_t	751
10.206.2.2graph_element	751
10.206.2.3P	752
10.206.3Constructor & Destructor Documentation	752
10.206.3.1graph_window_piter	752
10.206.3.2graph_window_piter	752
10.206.3.3graph_window_piter	752
10.206.4Member Function Documentation	752
10.206.4.1change_target_site_set	752
10.206.4.2element	753
10.206.4.3id	753
10.206.4.4next	753
10.206.4.5target_site_set	753
10.207ln::hexa< I > Struct Template Reference	753
10.207.1Detailed Description	755
10.207.2Member Typedef Documentation	755
10.207.2.1bkd_piter	755

10.207.2.2fwd_piter	755
10.207.2.3lvalue	755
10.207.2.4psite	756
10.207.2.5rvalue	756
10.207.2.6skeleton	756
10.207.2.7value	756
10.207.3.Constructor & Destructor Documentation	756
10.207.3.1hexa	756
10.207.3.2hexa	756
10.207.4.Member Function Documentation	756
10.207.4.1domain	756
10.207.4.2has	756
10.207.4.3operator()	757
10.207.4.4operator()	757
10.208.hln::histo::array< T > Struct Template Reference	757
10.208.Detailed Description	757
10.209.hln::Image< E > Struct Template Reference	757
10.209.Detailed Description	759
10.210.hln::image1d< T > Struct Template Reference	759
10.210.Detailed Description	760
10.210.2.Member Typedef Documentation	761
10.210.2.1lvalue	761
10.210.2.2rvalue	761
10.210.2.3skeleton	761
10.210.2.4value	761
10.210.3.Constructor & Destructor Documentation	761
10.210.3.1image1d	761
10.210.3.2image1d	761
10.210.3.3image1d	761
10.210.4.Member Function Documentation	762
10.210.4.1bbox	762
10.210.4.2border	762
10.210.4.3buffer	762
10.210.4.4buffer	762
10.210.4.5delta_index	762
10.210.4.6domain	762

10.210.4.7element	762
10.210.4.8element	763
10.210.4.9has	763
10.210.4.10elements	763
10.210.4.11inds	763
10.210.4.12operator()	763
10.210.4.13operator()	763
10.210.4.14point_at_index	763
10.210.4.15bbox	764
10.211ln::image2d< T > Class Template Reference	764
10.211.1Detailed Description	765
10.211.2Member Typedef Documentation	766
10.211.2.1lvalue	766
10.211.2.2rvalue	766
10.211.2.3skeleton	766
10.211.2.4value	766
10.211.3Constructor & Destructor Documentation	766
10.211.3.1image2d	766
10.211.3.2image2d	766
10.211.3.3image2d	766
10.211.4Member Function Documentation	767
10.211.4.1bbox	767
10.211.4.2border	767
10.211.4.3buffer	767
10.211.4.4buffer	767
10.211.4.5delta_index	767
10.211.4.6domain	767
10.211.4.7element	767
10.211.4.8element	768
10.211.4.9has	768
10.211.4.10cols	768
10.211.4.11elements	768
10.211.4.12rows	768
10.211.4.13operator()	768
10.211.4.14operator()	768
10.211.4.15point_at_index	769

10.212.1. <code>nl::image2d_h< V ></code> Struct Template Reference	769
10.212.2. Detailed Description	770
10.212.3. Member Typedef Documentation	770
10.212.2.1. <code>l_bkd_piter</code>	770
10.212.2.2. <code>2fwd_piter</code>	770
10.212.2.3. <code>lvalue</code>	771
10.212.2.4. <code>psite</code>	771
10.212.2.5. <code>rvalue</code>	771
10.212.2.6. <code>skeleton</code>	771
10.212.2.7. <code>value</code>	771
10.212.4. Constructor & Destructor Documentation	771
10.212.3.1. <code>image2d_h</code>	771
10.212.5. Member Function Documentation	771
10.212.4.1. <code>domain</code>	771
10.212.4.2. <code>has</code>	772
10.212.4.3. <code>operator()</code>	772
10.212.4.4. <code>operator()</code>	772
10.213.1. <code>nl::image3d< T ></code> Struct Template Reference	772
10.213.2. Detailed Description	774
10.213.3. Member Typedef Documentation	774
10.213.2.1. <code>lvalue</code>	774
10.213.2.2. <code>rvalue</code>	774
10.213.2.3. <code>skeleton</code>	774
10.213.2.4. <code>value</code>	774
10.213.4. Constructor & Destructor Documentation	775
10.213.3.1. <code>image3d</code>	775
10.213.3.2. <code>image3d</code>	775
10.213.3.3. <code>image3d</code>	775
10.213.5. Member Function Documentation	775
10.213.4.1. <code>l_bbox</code>	775
10.213.4.2. <code>border</code>	775
10.213.4.3. <code>buffer</code>	775
10.213.4.4. <code>buffer</code>	775
10.213.4.5. <code>delta_index</code>	776
10.213.4.6. <code>domain</code>	776
10.213.4.7. <code>element</code>	776

10.213.4.8element	776
10.213.4.9has	776
10.213.4.10cols	776
10.213.4.11elements	776
10.213.4.12rows	777
10.213.4.13slis	777
10.213.4.14operator()	777
10.213.4.15operator()	777
10.213.4.16point_at_index	777
10.213.4.17bbox	777
10.214.1Inn::interpolated< I, F > Struct Template Reference	777
10.214.1Detailed Description	778
10.214.2Member Typedef Documentation	778
10.214.2.1lvalue	778
10.214.2.2psite	779
10.214.2.3rvalue	779
10.214.2.4skeleton	779
10.214.2.5value	779
10.214.3Constructor & Destructor Documentation	779
10.214.3.1interpolated	779
10.214.4Member Function Documentation	779
10.214.4.1has	779
10.214.4.2is_valid	779
10.215.1Inn::io::dicom::dicom_header Struct Reference	780
10.215.1Detailed Description	780
10.216.1Inn::io::dump::dump_header Struct Reference	780
10.216.1Detailed Description	780
10.217.1Inn::io::fld::fld_header Struct Reference	780
10.217.1Detailed Description	780
10.218.1Inn::io::raw::raw_header Struct Reference	780
10.218.1Detailed Description	780
10.219.1Inn::Iterator< E > Struct Template Reference	781
10.219.1Detailed Description	783
10.219.2Member Function Documentation	783
10.219.2.1next	783
10.220.1Inn::labeled_image< I > Class Template Reference	783

10.220.Detailed Description	785
10.220.2Member Typedef Documentation	785
10.220.2.1bbox_t	785
10.220.2.2skeleton	785
10.220.3Constructor & Destructor Documentation	786
10.220.3.1labeled_image	786
10.220.3.2labeled_image	786
10.220.3.3labeled_image	786
10.220.4Member Function Documentation	786
10.220.4.1bbox	786
10.220.4.2bboxes	786
10.220.4.3nlabels	786
10.220.4.4relabel	786
10.220.4.5relabel	787
10.220.4.6subdomain	787
10.220.4.7update_data	787
10.221Inn::labeled_image_base< I, E > Class Template Reference	787
10.221.Detailed Description	788
10.221.2Member Typedef Documentation	789
10.221.2.1bbox_t	789
10.221.3Constructor & Destructor Documentation	789
10.221.3.1labeled_image_base	789
10.221.4Member Function Documentation	789
10.221.4.1bbox	789
10.221.4.2bboxes	789
10.221.4.3nlabels	789
10.221.4.4relabel	790
10.221.4.5relabel	790
10.221.4.6subdomain	790
10.221.4.7update_data	790
10.222Inn::lazy_image< I, F, B > Struct Template Reference	790
10.222.Detailed Description	791
10.222.2Member Typedef Documentation	792
10.222.2.1lvalue	792
10.222.2.2rvalue	792
10.222.2.3skeleton	792

10.222. Constructor & Destructor Documentation	792
10.222.3. lazy_image	792
10.222.3.2 lazy_image	792
10.222.4 Member Function Documentation	792
10.222.4.1 domain	792
10.222.4.2 has	792
10.222.4.3 operator()	793
10.222.4.4 operator()	793
10.222.4.5 operator()	793
10.222.4.6 operator()	793
10.223. literal< E > Struct Template Reference	793
10.223. Detailed Description	795
10.224. literal::black_t Struct Reference	795
10.224. Detailed Description	795
10.225. literal::blue_t Struct Reference	796
10.225. Detailed Description	796
10.226. literal::brown_t Struct Reference	796
10.226. Detailed Description	797
10.227. literal::cyan_t Struct Reference	797
10.227. Detailed Description	798
10.228. literal::green_t Struct Reference	798
10.228. Detailed Description	799
10.229. literal::identity_t Struct Reference	799
10.229. Detailed Description	800
10.230. literal::light_gray_t Struct Reference	800
10.230. Detailed Description	801
10.231. literal::lime_t Struct Reference	801
10.231. Detailed Description	802
10.232. literal::magenta_t Struct Reference	802
10.232. Detailed Description	803
10.233. literal::max_t Struct Reference	803
10.233. Detailed Description	804
10.234. literal::min_t Struct Reference	804
10.234. Detailed Description	805
10.235. literal::olive_t Struct Reference	805
10.235. Detailed Description	806

10.236	ln::literal::one_t Struct Reference	806
10.236	Detailed Description	807
10.237	ln::literal::orange_t Struct Reference	807
10.237	Detailed Description	808
10.238	ln::literal::origin_t Struct Reference	808
10.238	Detailed Description	809
10.239	ln::literal::pink_t Struct Reference	809
10.239	Detailed Description	810
10.240	ln::literal::purple_t Struct Reference	810
10.240	Detailed Description	811
10.241	ln::literal::red_t Struct Reference	811
10.241	Detailed Description	812
10.242	ln::literal::teal_t Struct Reference	812
10.242	Detailed Description	813
10.243	ln::literal::violet_t Struct Reference	813
10.243	Detailed Description	814
10.244	ln::literal::white_t Struct Reference	814
10.244	Detailed Description	815
10.245	ln::literal::yellow_t Struct Reference	815
10.245	Detailed Description	816
10.246	ln::literal::zero_t Struct Reference	816
10.246	Detailed Description	817
10.247	ln::Mesh< E > Struct Template Reference	817
10.247	Detailed Description	818
10.248	ln::Meta_Accumulator< E > Struct Template Reference	818
10.248	Detailed Description	819
10.249	ln::Meta_Function< E > Struct Template Reference	820
10.249	Detailed Description	820
10.250	ln::Meta_Function_v2v< E > Struct Template Reference	821
10.250	Detailed Description	821
10.251	ln::Meta_Function_vv2v< E > Struct Template Reference	821
10.251	Detailed Description	822
10.252	ln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference	822
10.252	Detailed Description	822
10.253	ln::metal::converts_to< T, U > Struct Template Reference	822
10.253	Detailed Description	822

10.254	<code>ln::metal::equal< T1, T2 ></code> Struct Template Reference	822
10.254	Detailed Description	823
10.255	<code>ln::metal::goes_to< T, U ></code> Struct Template Reference	823
10.255	Detailed Description	823
10.256	<code>ln::metal::is< T, U ></code> Struct Template Reference	823
10.256	Detailed Description	823
10.257	<code>ln::metal::is_a< T, M ></code> Struct Template Reference	824
10.257	Detailed Description	824
10.258	<code>ln::metal::is_not< T, U ></code> Struct Template Reference	824
10.258	Detailed Description	824
10.259	<code>ln::metal::is_not_a< T, M ></code> Struct Template Reference	824
10.259	Detailed Description	824
10.260	<code>ln::morpho::attribute::card< I ></code> Class Template Reference	824
10.260	Detailed Description	825
10.260	Member Function Documentation	825
10.260.2	<code>limit</code>	825
10.260.2	<code>is_valid</code>	825
10.260.2	<code>take_as_init</code>	825
10.260.2	<code>take_n_times</code>	826
10.260.2	<code>to_result</code>	826
10.261	<code>ln::morpho::attribute::count_adjacent_vertices< I ></code> Struct Template Reference	826
10.261	Detailed Description	826
10.261	Member Function Documentation	827
10.261.2	<code>limit</code>	827
10.261.2	<code>is_valid</code>	827
10.261.2	<code>take_as_init</code>	827
10.261.2	<code>take_n_times</code>	827
10.261.2	<code>to_result</code>	827
10.262	<code>ln::morpho::attribute::height< I ></code> Struct Template Reference	827
10.262	Detailed Description	828
10.262	Member Function Documentation	828
10.262.2	<code>lbase_level</code>	828
10.262.2	<code>init</code>	828
10.262.2	<code>is_valid</code>	828
10.262.2	<code>take_as_init</code>	829
10.262.2	<code>take_n_times</code>	829

10.262.2.6	to_result	829
10.263	ln::morpho::attribute::sharpness< I > Struct Template Reference	829
10.263.1	Detailed Description	830
10.263.2	Member Function Documentation	830
10.263.2.1	area	830
10.263.2.2	height	830
10.263.2.3	init	830
10.263.2.4	is_valid	830
10.263.2.5	take_as_init	831
10.263.2.6	take_n_times	831
10.263.2.7	to_result	831
10.263.2.8	volume	831
10.264	ln::morpho::attribute::sum< I, S > Class Template Reference	831
10.264.1	Detailed Description	832
10.264.2	Member Function Documentation	832
10.264.2.1	limit	832
10.264.2.2	is_valid	832
10.264.2.3	set_value	832
10.264.2.4	take_as_init	832
10.264.2.5	take_n_times	833
10.264.2.6	to_result	833
10.264.2.7	untake	833
10.265	ln::morpho::attribute::volume< I > Struct Template Reference	833
10.265.1	Detailed Description	834
10.265.2	Member Function Documentation	834
10.265.2.1	area	834
10.265.2.2	init	834
10.265.2.3	is_valid	834
10.265.2.4	take_as_init	834
10.265.2.5	take_n_times	834
10.265.2.6	to_result	834
10.266	ln::neighb< W > Class Template Reference	835
10.266.1	Detailed Description	835
10.266.2	Member Typedef Documentation	835
10.266.2.1	lbkd_niter	835
10.266.2.2	fwd_niter	836

10.266.2.3niter	836
10.266.3.Constructor & Destructor Documentation	836
10.266.3.1neighb	836
10.266.3.2neighb	836
10.267.mln::Neighborhood< E > Struct Template Reference	836
10.267.Detailed Description	837
10.268.mln::Neighborhood< void > Struct Template Reference	837
10.268.Detailed Description	837
10.269.mln::Object< E > Struct Template Reference	837
10.269.Detailed Description	837
10.270.mln::p2p_image< I, F > Struct Template Reference	837
10.270.Detailed Description	838
10.270.2.Member Typedef Documentation	838
10.270.2.1skeleton	838
10.270.3.Constructor & Destructor Documentation	839
10.270.3.1p2p_image	839
10.270.3.2p2p_image	839
10.270.4.Member Function Documentation	839
10.270.4.1domain	839
10.270.4.2fun	839
10.270.4.3operator()	839
10.270.4.4operator()	839
10.271.mln::p_array< P > Class Template Reference	839
10.271.Detailed Description	841
10.271.2.Member Typedef Documentation	841
10.271.2.1bkd_piter	841
10.271.2.2element	841
10.271.2.3fwd_piter	842
10.271.2.4element	842
10.271.2.5piter	842
10.271.2.6psite	842
10.271.3.Constructor & Destructor Documentation	842
10.271.3.1p_array	842
10.271.3.2p_array	842
10.271.4.Member Function Documentation	842
10.271.4.1append	842

10.271.4.2append	843
10.271.4.3change	843
10.271.4.4clear	843
10.271.4.5has	843
10.271.4.6has	843
10.271.4.7insert	843
10.271.4.8is_valid	843
10.271.4.9memory_size	844
10.271.4.10sites	844
10.271.4.11operator[]	844
10.271.4.12operator[]	844
10.271.4.13operator[]	844
10.271.4.14reserve	844
10.271.4.15size	845
10.271.4.16id_vector	845
10.272.mln::p_centered< W > Class Template Reference	845
10.272.Detailed Description	846
10.272.Member Typedef Documentation	846
10.272.2.lbkd_piter	846
10.272.2.2element	846
10.272.2.3fwd_piter	846
10.272.2.4piter	847
10.272.2.5psite	847
10.272.2.6site	847
10.272.3.Constructor & Destructor Documentation	847
10.272.3.1p_centered	847
10.272.3.2p_centered	847
10.272.4.Member Function Documentation	847
10.272.4.1center	847
10.272.4.2has	847
10.272.4.3is_valid	848
10.272.4.4memory_size	848
10.272.4.5window	848
10.273.mln::p_complex< D, G > Class Template Reference	848
10.273.Detailed Description	849
10.273.Member Typedef Documentation	849

10.273.2.1bkd_piter	849
10.273.2.2element	850
10.273.2.3fwd_piter	850
10.273.2.4piter	850
10.273.2.5psite	850
10.273.3.Constructor & Destructor Documentation	850
10.273.3.1p_complex	850
10.273.4.Member Function Documentation	850
10.273.4.1cplx	850
10.273.4.2cplx	851
10.273.4.3geom	851
10.273.4.4has	851
10.273.4.5is_valid	851
10.273.4.6faces	851
10.273.4.7nfaces_of_dim	851
10.273.4.8nsites	852
10.274.1.nn::p_edges< G, F > Class Template Reference	852
10.274.2.Detailed Description	854
10.274.3.Member Typedef Documentation	854
10.274.3.1bkd_piter	854
10.274.3.2edge	854
10.274.3.3element	854
10.274.3.4fun_t	854
10.274.3.5fwd_piter	854
10.274.3.6graph_element	855
10.274.3.7graph_t	855
10.274.3.8piter	855
10.274.3.9psite	855
10.274.4.Constructor & Destructor Documentation	855
10.274.4.1p_edges	855
10.274.4.2p_edges	855
10.274.4.3p_edges	856
10.274.4.4p_edges	856
10.274.5.Member Function Documentation	856
10.274.5.1function	856
10.274.5.2graph	856

10.274.4.3has	856
10.274.4.4has	857
10.274.4.5invalidate	857
10.274.4.6is_valid	857
10.274.4.7memory_size	857
10.274.4.8nedges	857
10.274.4.9nsites	857
10.275.mln::p_faces< N, D, P > Struct Template Reference	858
10.275.Detailed Description	859
10.275.Member Typedef Documentation	859
10.275.2.lbkd_piter	859
10.275.2.2element	859
10.275.2.3fwd_piter	859
10.275.2.4piter	859
10.275.2.5psite	859
10.275.Constructor & Destructor Documentation	859
10.275.3.lp_faces	859
10.275.3.2p_faces	860
10.275.Member Function Documentation	860
10.275.4.lcplx	860
10.275.4.2cplx	860
10.275.4.3is_valid	860
10.275.4.4nfaces	860
10.275.4.5nsites	861
10.276.mln::p_graph_piter< S, I > Class Template Reference	861
10.276.Detailed Description	861
10.276.Constructor & Destructor Documentation	862
10.276.2.lp_graph_piter	862
10.276.Member Function Documentation	862
10.276.3.lgraph	862
10.276.3.2d	862
10.276.3.3mln_q_subject	862
10.276.3.4next	862
10.277.mln::p_if< S, F > Class Template Reference	862
10.277.Detailed Description	864
10.277.Member Typedef Documentation	864

10.277.2.1bkd_piter	864
10.277.2.2element	864
10.277.2.3fwd_piter	864
10.277.2.4piter	864
10.277.2.5psite	864
10.277.3Constructor & Destructor Documentation	864
10.277.3.1p_if	864
10.277.3.2p_if	865
10.277.4Member Function Documentation	865
10.277.4.1has	865
10.277.4.2is_valid	865
10.277.4.3memory_size	865
10.277.4.4overset	865
10.277.4.5pred	865
10.277.4.6predicate	865
10.278.1.1p_image< I > Class Template Reference	866
10.278.2Detailed Description	867
10.278.3Member Typedef Documentation	867
10.278.3.1bkd_piter	867
10.278.3.2element	867
10.278.3.3fwd_piter	867
10.278.3.4element	868
10.278.3.5piter	868
10.278.3.6psite	868
10.278.3.7element	868
10.278.3.8S	868
10.278.4Constructor & Destructor Documentation	868
10.278.4.1p_image	868
10.278.4.2p_image	868
10.278.5Member Function Documentation	868
10.278.5.1clear	868
10.278.5.2has	869
10.278.5.3insert	869
10.278.5.4is_valid	869
10.278.5.5memory_size	869
10.278.5.6nsites	869

10.278.4.7operator typename internal::p_image_site_set< I >::ret	869
10.278.4.8remove	870
10.278.4.9toggle	870
10.279.1ln::p_indexed_bkd_piter< S > Class Template Reference	870
10.279.1.Detailed Description	870
10.279.1.Constructor & Destructor Documentation	871
10.279.2.1p_indexed_bkd_piter	871
10.279.2.2p_indexed_bkd_piter	871
10.279.3.Member Function Documentation	871
10.279.3.1index	871
10.279.3.2next	871
10.280.1ln::p_indexed_fwd_piter< S > Class Template Reference	871
10.280.1.Detailed Description	872
10.280.1.Constructor & Destructor Documentation	872
10.280.2.1p_indexed_fwd_piter	872
10.280.2.2p_indexed_fwd_piter	872
10.280.3.Member Function Documentation	872
10.280.3.1index	872
10.280.3.2next	872
10.281.1ln::p_indexed_psite< S > Class Template Reference	873
10.281.1.Detailed Description	873
10.282.1ln::p_key< K, P > Class Template Reference	873
10.282.1.Detailed Description	875
10.282.2.Member Typedef Documentation	875
10.282.2.1bkd_piter	875
10.282.2.2element	875
10.282.2.3fwd_piter	875
10.282.2.4_element	875
10.282.2.5piter	875
10.282.2.6psite	876
10.282.2.7_element	876
10.282.3.Constructor & Destructor Documentation	876
10.282.3.1p_key	876
10.282.4.Member Function Documentation	876
10.282.4.1change_key	876
10.282.4.2change_keys	876

10.282.4.3clear	876
10.282.4.4exists_key	876
10.282.4.5has	877
10.282.4.6has	877
10.282.4.7insert	877
10.282.4.8insert	877
10.282.4.9is_valid	877
10.282.4.10key	877
10.282.4.11keys	877
10.282.4.12memory_size	878
10.282.4.13sites	878
10.282.4.14operator()	878
10.282.4.15move	878
10.282.4.16move_key	878
10.283.1.1p_line2d Class Reference	878
10.283.2.1Detailed Description	880
10.283.2.2Member Typedef Documentation	880
10.283.2.3l_bkd_piter	880
10.283.2.4element	880
10.283.2.5fwd_piter	880
10.283.2.6piter	880
10.283.2.7psite	880
10.283.2.8q_box	880
10.283.3.1Constructor & Destructor Documentation	881
10.283.3.2lp_line2d	881
10.283.3.3p_line2d	881
10.283.4.1Member Function Documentation	881
10.283.4.2l_bbox	881
10.283.4.3begin	881
10.283.4.4end	881
10.283.4.5has	881
10.283.4.6has	882
10.283.4.7is_valid	882
10.283.4.8memory_size	882
10.283.4.9sites	882
10.283.4.10operator[]	882

10.283.4.1 ld_vector	882
10.284.1 ln::p_mutable_array_of< S > Class Template Reference	882
10.284.1 Detailed Description	884
10.284.2 Member Typedef Documentation	884
10.284.2.1 lkd_piter	884
10.284.2.2 element	884
10.284.2.3 fwd_piter	884
10.284.2.4 i_element	884
10.284.2.5 piter	884
10.284.2.6 psite	884
10.284.3 Constructor & Destructor Documentation	885
10.284.3.1 p_mutable_array_of	885
10.284.4 Member Function Documentation	885
10.284.4.1 clear	885
10.284.4.2 has	885
10.284.4.3 insert	885
10.284.4.4 is_valid	885
10.284.4.5 memory_size	885
10.284.4.6 elements	885
10.284.4.7 operator[]	886
10.284.4.8 operator[]	886
10.284.4.9 reserve	886
10.285.1 ln::p_n_faces_bkd_piter< D, G > Class Template Reference	886
10.285.1 Detailed Description	887
10.285.2 Constructor & Destructor Documentation	887
10.285.2.1 p_n_faces_bkd_piter	887
10.285.3 Member Function Documentation	887
10.285.3.1 ln	887
10.285.3.2 next	887
10.286.1 ln::p_n_faces_fwd_piter< D, G > Class Template Reference	887
10.286.1 Detailed Description	888
10.286.2 Constructor & Destructor Documentation	888
10.286.2.1 p_n_faces_fwd_piter	888
10.286.3 Member Function Documentation	888
10.286.3.1 ln	888
10.286.3.2 next	888

10.287.1. <code>std::priority_queue< P, Q ></code> Class Template Reference	889
10.287.2. Detailed Description	890
10.287.3. Member Typedef Documentation	890
10.287.4. <code>std::priority_queue::value_type</code>	890
10.287.5. <code>std::priority_queue::const_value_type</code>	891
10.287.6. <code>std::priority_queue::reference</code>	891
10.287.7. <code>std::priority_queue::const_reference</code>	891
10.287.8. <code>std::priority_queue::size_type</code>	891
10.287.9. <code>std::priority_queue::difference_type</code>	891
10.287.10. Constructor & Destructor Documentation	891
10.287.11. <code>std::priority_queue::priority_queue()</code>	891
10.287.12. Member Function Documentation	891
10.287.13. <code>std::priority_queue::clear()</code>	891
10.287.14. <code>std::priority_queue::exists()</code>	892
10.287.15. <code>std::priority_queue::front()</code>	892
10.287.16. <code>std::priority_queue::has()</code>	892
10.287.17. <code>std::priority_queue::highest_priority()</code>	892
10.287.18. <code>std::priority_queue::insert()</code>	892
10.287.19. <code>std::priority_queue::insert()</code>	893
10.287.20. <code>std::priority_queue::is_valid()</code>	893
10.287.21. <code>std::priority_queue::lowest_priority()</code>	893
10.287.22. <code>std::priority_queue::memory_size()</code>	893
10.287.23. <code>std::priority_queue::sites()</code>	893
10.287.24. <code>std::priority_queue::operator()</code>	893
10.287.25. <code>std::priority_queue::pop()</code>	894
10.287.26. <code>std::priority_queue::pop_front()</code>	894
10.287.27. <code>std::priority_queue::priorities()</code>	894
10.287.28. <code>std::priority_queue::push()</code>	894
10.288.1. <code>std::queue< P ></code> Class Template Reference	894
10.288.2. Detailed Description	896
10.288.3. Member Typedef Documentation	896
10.288.4. <code>std::queue::value_type</code>	896
10.288.5. <code>std::queue::const_value_type</code>	896
10.288.6. <code>std::queue::reference</code>	896
10.288.7. <code>std::queue::const_reference</code>	896
10.288.8. <code>std::queue::size_type</code>	896
10.288.9. <code>std::queue::difference_type</code>	897

10.288.2.6psite	897
10.288.3.Constructor & Destructor Documentation	897
10.288.3.1p_queue	897
10.288.4.Member Function Documentation	897
10.288.4.1clear	897
10.288.4.2front	897
10.288.4.3has	897
10.288.4.4has	897
10.288.4.5insert	898
10.288.4.6is_valid	898
10.288.4.7memory_size	898
10.288.4.8sites	898
10.288.4.9operator[]	898
10.288.4.10pop	898
10.288.4.11pop_front	898
10.288.4.12push	899
10.288.4.13rd_deque	899
10.289.1.1p_queue_fast< P > Class Template Reference	899
10.289.2.Detailed Description	901
10.289.3.Member Typedef Documentation	901
10.289.3.1bkd_piter	901
10.289.3.2element	901
10.289.3.3fwd_piter	901
10.289.3.4_element	901
10.289.3.5piter	901
10.289.3.6psite	902
10.289.4.Constructor & Destructor Documentation	902
10.289.4.1p_queue_fast	902
10.289.5.Member Function Documentation	902
10.289.5.1clear	902
10.289.5.2compute_has	902
10.289.5.3empty	902
10.289.5.4front	902
10.289.5.5has	902
10.289.5.6has	903
10.289.5.7insert	903

10.289.4.8s_valid	903
10.289.4.9memory_size	903
10.289.4.10sites	903
10.289.4.11operator[]	903
10.289.4.12op	903
10.289.4.13op_front	904
10.289.4.14purge	904
10.289.4.15push	904
10.289.4.16serve	904
10.289.4.17td_vector	904
10.290.1In::p_run< P > Class Template Reference	904
10.290.2Detailed Description	906
10.290.3Member Typedef Documentation	906
10.290.2.1bkd_piter	906
10.290.2.2element	906
10.290.2.3fwd_piter	906
10.290.2.4piter	906
10.290.2.5psite	907
10.290.2.6q_box	907
10.290.3Constructor & Destructor Documentation	907
10.290.3.1p_run	907
10.290.3.2p_run	907
10.290.3.3p_run	907
10.290.4Member Function Documentation	907
10.290.4.1bbox	907
10.290.4.2end	907
10.290.4.3has	908
10.290.4.4has	908
10.290.4.5has_index	908
10.290.4.6nit	908
10.290.4.7s_valid	908
10.290.4.8length	908
10.290.4.9memory_size	909
10.290.4.10sites	909
10.290.4.11operator[]	909
10.290.4.12art	909

10.291. <code>ln::p_set< P ></code> Class Template Reference	909
10.291.1. Detailed Description	911
10.291.2. Member Typedef Documentation	911
10.291.2.1. <code>l_bkd_piter</code>	911
10.291.2.2. <code>element</code>	911
10.291.2.3. <code>fwd_piter</code>	911
10.291.2.4. <code>element</code>	911
10.291.2.5. <code>piter</code>	911
10.291.2.6. <code>psite</code>	911
10.291.2.7. <code>element</code>	911
10.291.3. Constructor & Destructor Documentation	912
10.291.3.1. <code>lp_set</code>	912
10.291.4. Member Function Documentation	912
10.291.4.1. <code>clear</code>	912
10.291.4.2. <code>has</code>	912
10.291.4.3. <code>has</code>	912
10.291.4.4. <code>has</code>	912
10.291.4.5. <code>insert</code>	912
10.291.4.6. <code>is_valid</code>	912
10.291.4.7. <code>memory_size</code>	913
10.291.4.8. <code>nsites</code>	913
10.291.4.9. <code>operator[]</code>	913
10.291.4.10. <code>move</code>	913
10.291.4.11. <code>std_vector</code>	913
10.291.4.12. <code>til_set</code>	913
10.292. <code>ln::p_transformed< S, F ></code> Class Template Reference	913
10.292.1. Detailed Description	914
10.292.2. Member Typedef Documentation	915
10.292.2.1. <code>l_bkd_piter</code>	915
10.292.2.2. <code>element</code>	915
10.292.2.3. <code>fwd_piter</code>	915
10.292.2.4. <code>piter</code>	915
10.292.2.5. <code>psite</code>	915
10.292.3. Constructor & Destructor Documentation	915
10.292.3.1. <code>lp_transformed</code>	915
10.292.3.2. <code>lp_transformed</code>	915

10.292.4	Member Function Documentation	916
10.292.4.1	function	916
10.292.4.2	has	916
10.292.4.3	is_valid	916
10.292.4.4	memory_size	916
10.292.4.5	primary_set	916
10.293	nl::p_transformed_piter< Pi, S, F > Struct Template Reference	916
10.293.1	Detailed Description	917
10.293.2	Constructor & Destructor Documentation	917
10.293.2.1	lp_transformed_piter	917
10.293.2.2	pp_transformed_piter	917
10.293.3	Member Function Documentation	917
10.293.3.1	lchange_target	917
10.293.3.2	next	918
10.294	nl::p_vaccess< V, S > Class Template Reference	918
10.294.1	Detailed Description	919
10.294.2	Member Typedef Documentation	919
10.294.2.1	lbgd_piter	919
10.294.2.2	element	920
10.294.2.3	fwd_piter	920
10.294.2.4	element	920
10.294.2.5	piter	920
10.294.2.6	pset	920
10.294.2.7	psite	920
10.294.2.8	value	920
10.294.2.9	vset	920
10.294.3	Constructor & Destructor Documentation	921
10.294.3.1	lp_vaccess	921
10.294.4	Member Function Documentation	921
10.294.4.1	lhas	921
10.294.4.2	has	921
10.294.4.3	insert	921
10.294.4.4	insert	921
10.294.4.5	is_valid	921
10.294.4.6	memory_size	921
10.294.4.7	operator()	922

10.294.4.8	values	922
10.295.1	mln::p_vertices< G, F > Class Template Reference	922
10.295.1	Detailed Description	924
10.295.2	Member Typedef Documentation	924
10.295.2.1	l_bkd_piter	924
10.295.2.2	element	924
10.295.2.3	fun_t	924
10.295.2.4	wd_piter	924
10.295.2.5	graph_element	925
10.295.2.6	graph_t	925
10.295.2.7	piter	925
10.295.2.8	psite	925
10.295.2.9	vertex	925
10.295.3	Constructor & Destructor Documentation	925
10.295.3.1	lp_vertices	925
10.295.3.2	2p_vertices	925
10.295.3.3	3p_vertices	926
10.295.3.4	4p_vertices	926
10.295.3.5	5p_vertices	926
10.295.4	Member Function Documentation	926
10.295.4.1	function	926
10.295.4.2	graph	927
10.295.4.3	has	927
10.295.4.4	has	927
10.295.4.5	invalidate	927
10.295.4.6	is_valid	927
10.295.4.7	memory_size	927
10.295.4.8	nsites	928
10.295.4.9	nvertices	928
10.295.4.10	operator()	928
10.296.1	mln::pixel< I > Struct Template Reference	928
10.296.1	Detailed Description	929
10.296.2	Constructor & Destructor Documentation	929
10.296.2.1	lpixel	929
10.296.2.2	2pixel	929
10.296.3	Member Function Documentation	929

10.296.3.lchange_to	929
10.296.3.2s_valid	930
10.297.nln::plain< I > Class Template Reference	930
10.297.1.Detailed Description	930
10.297.2.Member Typedef Documentation	931
10.297.2.1.skeleton	931
10.297.3.Constructor & Destructor Documentation	931
10.297.3.1.plain	931
10.297.3.2.plain	931
10.297.3.3.plain	931
10.297.4.Member Function Documentation	931
10.297.4.1.operator I	931
10.297.4.2.operator=	931
10.297.4.3.operator=	931
10.298.nln::Point< P > Struct Template Reference	932
10.298.1.Detailed Description	932
10.298.2.Member Typedef Documentation	933
10.298.2.1.point	933
10.298.3.Member Function Documentation	933
10.298.3.1.to_point	933
10.298.4.Friends And Related Function Documentation	933
10.298.4.1.operator+=	933
10.298.4.2.operator-=	933
10.298.4.3.operator/	934
10.299.nln::point< G, C > Struct Template Reference	934
10.299.1.Detailed Description	937
10.299.2.Member Typedef Documentation	937
10.299.2.1.coord	937
10.299.2.2.delta	937
10.299.2.3.dpsite	937
10.299.2.4.grid	937
10.299.2.5.h_vec	938
10.299.2.6.vec	938
10.299.3.Member Enumeration Documentation	938
10.299.3.1."@30	938
10.299.4.Constructor & Destructor Documentation	938

10.299.4.1point	938
10.299.4.2point	938
10.299.4.3point	938
10.299.4.4point	938
10.299.4.5point	939
10.299.5Member Function Documentation	939
10.299.5.1last_coord	939
10.299.5.2last_coord	939
10.299.5.3minus_infty	939
10.299.5.4operator+=	939
10.299.5.5operator-=	939
10.299.5.6operator[]	939
10.299.5.7operator[]	940
10.299.5.8plus_infty	940
10.299.5.9set_all	940
10.299.5.10h_vec	940
10.299.5.11b_vec	940
10.299.6Member Data Documentation	941
10.299.6.1origin	941
10.300Inn::Proxy< E > Struct Template Reference	941
10.300.Detailed Description	941
10.301Inn::Proxy< void > Struct Template Reference	941
10.301.Detailed Description	941
10.302Inn::Pseudo_Site< E > Struct Template Reference	941
10.302.Detailed Description	942
10.303Inn::Pseudo_Site< void > Struct Template Reference	942
10.303.Detailed Description	942
10.304Inn::pw::image< F, S > Class Template Reference	942
10.304.Detailed Description	943
10.304.2Member Typedef Documentation	943
10.304.2.1skeleton	943
10.304.3Constructor & Destructor Documentation	943
10.304.3.1image	943
10.304.3.2image	943
10.305Inn::registration::closest_point_basic< P > Class Template Reference	943
10.305.Detailed Description	944

10.306	nl::registration::closest_point_with_map< P > Class Template Reference	944
10.306	Detailed Description	944
10.307	nl::Regular_Grid< E > Struct Template Reference	944
10.307	Detailed Description	944
10.308	nl::safe_image< I > Class Template Reference	945
10.308	Detailed Description	945
10.308	Member Typedef Documentation	945
10.308.2	Iskeleton	945
10.308	Member Function Documentation	945
10.308.3	operator safe_image< const I >	945
10.309	nl::select::p_of< P > Struct Template Reference	945
10.309	Detailed Description	946
10.310	nl::Site< E > Struct Template Reference	946
10.310	Detailed Description	946
10.311	nl::Site< void > Struct Template Reference	947
10.311	Detailed Description	947
10.312	nl::Site_Iterator< E > Struct Template Reference	947
10.312	Detailed Description	949
10.312	Member Function Documentation	949
10.312.2	Inext	949
10.313	nl::Site_Proxy< E > Struct Template Reference	949
10.313	Detailed Description	949
10.314	nl::Site_Proxy< void > Struct Template Reference	950
10.314	Detailed Description	950
10.315	nl::Site_Set< E > Struct Template Reference	950
10.315	Detailed Description	951
10.315	Friends And Related Function Documentation	952
10.315.2	ldiff	952
10.315.2	2inter	952
10.315.2	3operator<	952
10.315.2	4operator<<	952
10.315.2	5operator<=	952
10.315.2	6operator==	953
10.315.2	7sym_diff	953
10.315.2	8uni	953
10.315.2	9unique	953

10.316	<code>ln::Site_Set< void ></code> Struct Template Reference	953
10.316.	Detailed Description	953
10.317	<code>ln::sub_image< I, S ></code> Class Template Reference	953
10.317.	Detailed Description	954
10.317.	Member Typedef Documentation	954
10.317.2.	<code>lskeleton</code>	954
10.317.	Constructor & Destructor Documentation	954
10.317.3.	<code>lsub_image</code>	954
10.317.3.	<code>sub_image</code>	955
10.317.	Member Function Documentation	955
10.317.4.	<code>ldomain</code>	955
10.317.4.	<code>operator sub_image< const I, S ></code>	955
10.318	<code>ln::sub_image_if< I, S ></code> Struct Template Reference	955
10.318.	Detailed Description	956
10.318.	Member Typedef Documentation	956
10.318.2.	<code>lskeleton</code>	956
10.318.	Constructor & Destructor Documentation	956
10.318.3.	<code>lsub_image_if</code>	956
10.318.3.	<code>sub_image_if</code>	956
10.318.	Member Function Documentation	956
10.318.4.	<code>ldomain</code>	956
10.319	<code>ln::thru_image< I, F ></code> Class Template Reference	956
10.319.	Detailed Description	957
10.319.	Member Function Documentation	957
10.319.2.	<code>operator thru_image< const I, F ></code>	957
10.320	<code>ln::thrubin_image< I1, I2, F ></code> Class Template Reference	957
10.320.	Detailed Description	958
10.320.	Member Typedef Documentation	958
10.320.2.	<code>lpsite</code>	958
10.320.2.	<code>rvalue</code>	958
10.320.2.	<code>skeleton</code>	958
10.320.2.	<code>value</code>	958
10.320.	Member Function Documentation	958
10.320.3.	<code>operator thrubin_image< const I1, const I2, F ></code>	958
10.321	<code>ln::topo::adj_higher_dim_connected_n_face_bkd_iter< D ></code> Class Template Reference	959
10.321.	Detailed Description	959

10.321. Constructor & Destructor Documentation	959
10.321.2. ladj_higher_dim_connected_n_face_bkd_iter	959
10.321. Member Function Documentation	959
10.321.3. lnext	959
10.322. mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference	960
10.322. Detailed Description	960
10.322. Constructor & Destructor Documentation	960
10.322.2. ladj_higher_dim_connected_n_face_fwd_iter	960
10.322. Member Function Documentation	961
10.322.3. lnext	961
10.323. mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference	961
10.323. Detailed Description	961
10.323. Constructor & Destructor Documentation	962
10.323.2. ladj_higher_face_bkd_iter	962
10.323. Member Function Documentation	962
10.323.3. lnext	962
10.324. mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference	962
10.324. Detailed Description	962
10.324. Constructor & Destructor Documentation	963
10.324.2. ladj_higher_face_fwd_iter	963
10.324. Member Function Documentation	963
10.324.3. lnext	963
10.325. mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference	963
10.325. Detailed Description	964
10.325. Constructor & Destructor Documentation	964
10.325.2. ladj_lower_dim_connected_n_face_bkd_iter	964
10.325. Member Function Documentation	964
10.325.3. lnext	964
10.326. mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference	964
10.326. Detailed Description	965
10.326. Constructor & Destructor Documentation	965
10.326.2. ladj_lower_dim_connected_n_face_fwd_iter	965
10.326. Member Function Documentation	965
10.326.3. lnext	965
10.327. mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference	965
10.327. Detailed Description	966

10.327.	Constructor & Destructor Documentation	966
10.327.2.	ladj_lower_face_bkd_iter	966
10.327.	Member Function Documentation	966
10.327.3.	lnext	966
10.328.	ln::topo::adj_lower_face_fwd_iter< D > Class Template Reference	967
10.328.	Detailed Description	967
10.328.	Constructor & Destructor Documentation	967
10.328.2.	ladj_lower_face_fwd_iter	967
10.328.	Member Function Documentation	967
10.328.3.	lnext	967
10.329.	ln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference	968
10.329.	Detailed Description	968
10.329.	Constructor & Destructor Documentation	968
10.329.2.	ladj_lower_higher_face_bkd_iter	968
10.329.	Member Function Documentation	968
10.329.3.	lnext	968
10.330.	ln::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference	969
10.330.	Detailed Description	969
10.330.	Constructor & Destructor Documentation	969
10.330.2.	ladj_lower_higher_face_fwd_iter	969
10.330.	Member Function Documentation	970
10.330.3.	lnext	970
10.331.	ln::topo::adj_m_face_bkd_iter< D > Class Template Reference	970
10.331.	Detailed Description	970
10.331.	Constructor & Destructor Documentation	971
10.331.2.	ladj_m_face_bkd_iter	971
10.331.2.	2adj_m_face_bkd_iter	971
10.331.	Member Function Documentation	971
10.331.3.	lnext	971
10.332.	ln::topo::adj_m_face_fwd_iter< D > Class Template Reference	971
10.332.	Detailed Description	972
10.332.	Constructor & Destructor Documentation	972
10.332.2.	ladj_m_face_fwd_iter	972
10.332.2.	2adj_m_face_fwd_iter	972
10.332.	Member Function Documentation	972
10.332.3.	lnext	972

10.333.1. <code>ln::topo::algebraic_face< D ></code> Class Template Reference	973
10.333.2. Detailed Description	974
10.333.3. Constructor & Destructor Documentation	975
10.333.4. <code>algebraic_face</code>	975
10.333.5. <code>algebraic_face</code>	975
10.333.6. <code>algebraic_face</code>	975
10.333.7. <code>algebraic_face</code>	975
10.333.8. Member Function Documentation	975
10.333.9. <code>lcplx</code>	975
10.333.10. <code>data</code>	975
10.333.11. <code>dec_face_id</code>	976
10.333.12. <code>dec_n</code>	976
10.333.13. <code>face_id</code>	976
10.333.14. <code>higher_dim_adj_faces</code>	976
10.333.15. <code>inc_face_id</code>	976
10.333.16. <code>inc_n</code>	976
10.333.17. <code>invalidate</code>	976
10.333.18. <code>is_valid</code>	977
10.333.19. <code>lower_dim_adj_faces</code>	977
10.333.20. <code>l2</code>	977
10.333.21. <code>set_cplx</code>	977
10.333.22. <code>set_face_id</code>	977
10.333.23. <code>set_n</code>	977
10.333.24. <code>set_sign</code>	977
10.333.25. <code>sign</code>	978
10.334.1. <code>ln::topo::algebraic_n_face< N, D ></code> Class Template Reference	978
10.334.2. Detailed Description	979
10.334.3. Constructor & Destructor Documentation	980
10.334.4. <code>algebraic_n_face</code>	980
10.334.5. <code>algebraic_n_face</code>	980
10.334.6. <code>algebraic_n_face</code>	980
10.334.7. Member Function Documentation	980
10.334.8. <code>lcplx</code>	980
10.334.9. <code>data</code>	980
10.334.10. <code>dec_face_id</code>	980
10.334.11. <code>face_id</code>	981

10.334.3.5higher_dim_adj_faces	981
10.334.3.6inc_face_id	981
10.334.3.7invalidate	981
10.334.3.8is_valid	981
10.334.3.9lower_dim_adj_faces	981
10.334.3.10	982
10.334.3.11set_cplx	982
10.334.3.12set_face_id	982
10.334.3.13set_sign	982
10.334.3.14sign	982
10.335.mln::topo::center_only_iter< D > Class Template Reference	982
10.335.1Detailed Description	983
10.335.2Constructor & Destructor Documentation	983
10.335.2.1center_only_iter	983
10.335.3Member Function Documentation	983
10.335.3.1next	983
10.336.mln::topo::centered_bkd_iter_adapter< D, I > Class Template Reference	984
10.336.1Detailed Description	984
10.336.2Constructor & Destructor Documentation	984
10.336.2.1centered_bkd_iter_adapter	984
10.336.3Member Function Documentation	984
10.336.3.1next	984
10.337.mln::topo::centered_fwd_iter_adapter< D, I > Class Template Reference	985
10.337.1Detailed Description	985
10.337.2Constructor & Destructor Documentation	985
10.337.2.1centered_fwd_iter_adapter	985
10.337.3Member Function Documentation	986
10.337.3.1next	986
10.338.mln::topo::complex< D > Class Template Reference	986
10.338.1Detailed Description	987
10.338.2Member Typedef Documentation	987
10.338.2.1bkd_citer	987
10.338.2.2fwd_citer	987
10.338.3Constructor & Destructor Documentation	987
10.338.3.1complex	987
10.338.4Member Function Documentation	988

10.338.4.1add_face	988
10.338.4.2add_face	988
10.338.4.3addr	988
10.338.4.4nfaces	988
10.338.4.5nfaces_of_dim	988
10.338.4.6nfaces_of_static_dim	989
10.338.4.7print	989
10.338.4.8print_faces	989
10.339.1n::topo::face< D > Class Template Reference	989
10.339.2Detailed Description	991
10.339.3Constructor & Destructor Documentation	991
10.339.4.1face	991
10.339.4.2face	991
10.339.4.3face	991
10.339.5Member Function Documentation	991
10.339.5.1cplx	991
10.339.5.2data	991
10.339.5.3dec_face_id	992
10.339.5.4dec_n	992
10.339.5.5face_id	992
10.339.5.6higher_dim_adj_faces	992
10.339.5.7inc_face_id	992
10.339.5.8inc_n	992
10.339.5.9invalidate	992
10.339.5.10valid	992
10.339.5.11bwer_dim_adj_faces	993
10.339.5.12	993
10.339.5.13set_cplx	993
10.339.5.14set_face_id	993
10.339.5.15set_n	993
10.340.1n::topo::face_bkd_iter< D > Class Template Reference	993
10.340.2Detailed Description	994
10.340.3Constructor & Destructor Documentation	994
10.340.4.1face_bkd_iter	994
10.340.5Member Function Documentation	994
10.340.5.1next	994

10.340.3.2start	994
10.341ln::topo::face_fwd_iter< D > Class Template Reference	995
10.341.Detailed Description	995
10.341.Constructor & Destructor Documentation	995
10.341.2.lface_fwd_iter	995
10.341.3.Member Function Documentation	995
10.341.3.lnext	995
10.341.3.2start	996
10.342ln::topo::is_n_face< N > Struct Template Reference	996
10.342.Detailed Description	997
10.343ln::topo::is_simple_2d_t< N > Struct Template Reference	997
10.343.Detailed Description	998
10.344ln::topo::is_simple_cell< I > Class Template Reference	998
10.344.Detailed Description	1000
10.344.2.Member Typedef Documentation	1000
10.344.2.lpsite	1000
10.344.2.2result	1000
10.344.3.Member Function Documentation	1000
10.344.3.lmln_geom	1000
10.344.3.2operator()	1000
10.344.3.3set_image	1001
10.344.4.Member Data Documentation	1001
10.344.4.ID	1001
10.345ln::topo::n_face< N, D > Class Template Reference	1001
10.345.Detailed Description	1002
10.345.Constructor & Destructor Documentation	1002
10.345.2.ln_face	1002
10.345.2.2n_face	1003
10.345.3.Member Function Documentation	1003
10.345.3.lcplx	1003
10.345.3.2data	1003
10.345.3.3dec_face_id	1003
10.345.3.4face_id	1003
10.345.3.5higher_dim_adj_faces	1003
10.345.3.6nc_face_id	1004
10.345.3.7invalidate	1004

10.345.3.8s_valid	1004
10.345.3.9lower_dim_adj_faces	1004
10.345.3.10	1004
10.345.3.1set_cplx	1004
10.345.3.1set_face_id	1004
10.346ln::topo::n_face_bkd_iter< D > Class Template Reference	1005
10.346.Detailed Description	1005
10.346.Member Function Documentation	1005
10.346.2.ln	1005
10.346.2.2next	1006
10.346.2.3start	1006
10.347ln::topo::n_face_fwd_iter< D > Class Template Reference	1006
10.347.Detailed Description	1006
10.347.Member Function Documentation	1007
10.347.2.ln	1007
10.347.2.2next	1007
10.347.2.3start	1007
10.348ln::topo::n_faces_set< N, D > Class Template Reference	1007
10.348.Detailed Description	1008
10.348.Member Typedef Documentation	1008
10.348.2.lfaces_type	1008
10.348.Member Function Documentation	1008
10.348.3.ladd	1008
10.348.3.2faces	1008
10.348.3.3reserve	1009
10.349ln::topo::skeleton::is_simple_point< N > Struct Template Reference	1009
10.349.Detailed Description	1009
10.350ln::topo::static_n_face_bkd_iter< N, D > Class Template Reference	1009
10.350.Detailed Description	1010
10.350.Constructor & Destructor Documentation	1010
10.350.2.lstatic_n_face_bkd_iter	1010
10.350.Member Function Documentation	1010
10.350.3.lnext	1010
10.350.3.2start	1010
10.351ln::topo::static_n_face_fwd_iter< N, D > Class Template Reference	1011
10.351.Detailed Description	1011

10.351.2	Constructor & Destructor Documentation	1011
10.351.2.1	static_n_face_fwd_iter	1011
10.351.3	Member Function Documentation	1011
10.351.3.1	next	1011
10.351.3.2	start	1012
10.352	nl::tr_image< S, I, T > Struct Template Reference	1012
10.352.1	Detailed Description	1013
10.352.2	Member Typedef Documentation	1013
10.352.2.1	lvalue	1013
10.352.2.2	psite	1013
10.352.2.3	rvalue	1013
10.352.2.4	site	1014
10.352.2.5	skeleton	1014
10.352.2.6	value	1014
10.352.3	Constructor & Destructor Documentation	1014
10.352.3.1	tr_image	1014
10.352.4	Member Function Documentation	1014
10.352.4.1	domain	1014
10.352.4.2	has	1014
10.352.4.3	is_valid	1014
10.352.4.4	operator()	1015
10.352.4.5	set_tr	1015
10.352.4.6	tr	1015
10.353	nl::transformed_image< I, F > Struct Template Reference	1015
10.353.1	Detailed Description	1016
10.353.2	Member Typedef Documentation	1016
10.353.2.1	lskeleton	1016
10.353.3	Constructor & Destructor Documentation	1016
10.353.3.1	ltransformed_image	1016
10.353.3.2	transformed_image	1016
10.353.4	Member Function Documentation	1016
10.353.4.1	domain	1016
10.353.4.2	operator transformed_image< const I, F >	1017
10.353.4.3	operator()	1017
10.353.4.4	operator()	1017
10.354	nl::unproject_image< I, D, F > Struct Template Reference	1017

10.354. Detailed Description	1018
10.354. Constructor & Destructor Documentation	1018
10.354.2. lunproject_image	1018
10.354.2. lunproject_image	1018
10.354. Member Function Documentation	1018
10.354.3. ldomain	1018
10.354.3. loperator()	1018
10.354.3. loperator()	1018
10.355. mln::util::adjacency_matrix< V > Class Template Reference	1019
10.355. Detailed Description	1019
10.355. Constructor & Destructor Documentation	1019
10.355.2. ladjacency_matrix	1019
10.355.2. ladjacency_matrix	1019
10.356. mln::util::array< T > Class Template Reference	1019
10.356. Detailed Description	1022
10.356. Member Typedef Documentation	1022
10.356.2. lbk_d_eter	1022
10.356.2. l_eter	1022
10.356.2. lelement	1022
10.356.2. lfwd_eter	1022
10.356.2. lresult	1023
10.356. Constructor & Destructor Documentation	1023
10.356.3. larray	1023
10.356.3. larray	1023
10.356.3. larray	1023
10.356. Member Function Documentation	1023
10.356.4. lappend	1023
10.356.4. lappend	1023
10.356.4. lclear	1024
10.356.4. lfill	1024
10.356.4. lis_empty	1024
10.356.4. llast	1024
10.356.4. llast	1024
10.356.4. lmemory_size	1024
10.356.4. lelements	1025
10.356.4. loperator()	1025

cxi		CONTENTS
	10.356.4.loperator()	102
	10.356.4.loperator[]	102
	10.356.4.loperator[]	102
	10.356.4.lreserve	102
	10.356.4.lsize	102
	10.356.4.lsize	102
	10.356.4.lsize	102
	10.356.4.lstd_vector	102
10.357	util::branch< T > Class Template Reference	102
	10.357.Detailed Description	102
	10.357.Constructor & Destructor Documentation	102
	10.357.2.branch	102
	10.357.3.Member Function Documentation	102
	10.357.3.lapex	102
	10.357.3.2.util_tree	102
10.358	util::branch_iter< T > Class Template Reference	102
	10.358.Detailed Description	102
	10.358.2.Member Function Documentation	102
	10.358.2.ldeepness	102
	10.358.2.2.invalidate	102
	10.358.2.3.is_valid	102
	10.358.2.4.next	102
	10.358.2.5.operator util::tree_node< T > &	102
	10.358.2.6.start	103
10.359	util::branch_iter_ind< T > Class Template Reference	103
	10.359.Detailed Description	103
	10.359.2.Member Function Documentation	103
	10.359.2.ldeepness	103
	10.359.2.2.invalidate	103
	10.359.2.3.is_valid	103
	10.359.2.4.next	103
	10.359.2.5.operator util::tree_node< T > &	103
	10.359.2.6.start	103
10.360	util::couple< T, U > Class Template Reference	103
	10.360.Detailed Description	103
	10.360.2.Member Function Documentation	103

10.360.2.1change_both	1033
10.360.2.2change_first	1033
10.360.2.3change_second	1033
10.360.2.4first	1033
10.360.2.5second	1033
10.361In::util::eat Struct Reference	1033
10.361.Detailed Description	1034
10.362In::util::edge< G > Class Template Reference	1034
10.362.Detailed Description	1035
10.362.Member Typedef Documentation	1036
10.362.2.1category	1036
10.362.2.2graph_t	1036
10.362.2.3id_t	1036
10.362.2.4id_value_t	1036
10.362.Constructor & Destructor Documentation	1036
10.362.3.1edge	1036
10.362.Member Function Documentation	1036
10.362.4.1change_graph	1036
10.362.4.2graph	1036
10.362.4.3id	1037
10.362.4.4invalidate	1037
10.362.4.5is_valid	1037
10.362.4.6th_nbh_edge	1037
10.362.4.7nmax_nbh_edges	1037
10.362.4.8operator edge_id_t	1037
10.362.4.9update_id	1037
10.362.4.10l	1038
10.362.4.11l2	1038
10.362.4.12l_other	1038
10.363In::util::fibonacci_heap< P, T > Class Template Reference	1038
10.363.Detailed Description	1040
10.363.Constructor & Destructor Documentation	1040
10.363.2.1fibonacci_heap	1040
10.363.2.2fibonacci_heap	1040
10.363.Member Function Documentation	1040
10.363.3.1clear	1040

10.363.3.2front	1040
10.363.3.3is_empty	1041
10.363.3.4is_valid	1041
10.363.3.5elements	1041
10.363.3.6operator=	1041
10.363.3.7pop_front	1041
10.363.3.8push	1041
10.363.3.9push	1042
10.364.1.1util::graph Class Reference	1042
10.364.2.1Detailed Description	1044
10.364.2.2Member Typedef Documentation	1044
10.364.2.2.1edge_fwd_iter	1044
10.364.2.2.2edge_nbh_edge_fwd_iter	1044
10.364.2.2.3edges_set_t	1044
10.364.2.2.4edges_t	1044
10.364.2.2.5vertex_nbh_edge_fwd_iter	1044
10.364.2.2.6vertex_nbh_vertex_fwd_iter	1045
10.364.2.2.7vertices_t	1045
10.364.2.3Constructor & Destructor Documentation	1045
10.364.2.3.1lgraph	1045
10.364.2.3.2graph	1045
10.364.2.4Member Function Documentation	1045
10.364.2.4.1add_edge	1045
10.364.2.4.2add_vertex	1045
10.364.2.4.3add_vertices	1046
10.364.2.4.4e_ith_nbh_edge	1046
10.364.2.4.5e_nmax	1046
10.364.2.4.6edge	1046
10.364.2.4.7edge	1046
10.364.2.4.8edges	1047
10.364.2.4.9has_e	1047
10.364.2.4.10has_v	1047
10.364.2.4.11is_subgraph_of	1047
10.364.2.4.12l	1047
10.364.2.4.13l2	1047
10.364.2.4.14l_ith_nbh_edge	1047

10.364.4.15_ith_nbh_vertex	1048
10.364.4.16_nmax	1048
10.364.4.17_vertex	1048
10.365.1ln::util::greater_point< I > Class Template Reference	1048
10.365.1.Detailed Description	1048
10.365.2.Member Function Documentation	1049
10.365.2.1.operator()	1049
10.366.1ln::util::greater_site< I > Class Template Reference	1049
10.366.1.Detailed Description	1049
10.366.2.Member Function Documentation	1049
10.366.2.1.operator()	1049
10.367.1ln::util::head< T, R > Class Template Reference	1049
10.367.1.Detailed Description	1050
10.368.1ln::util::ignore Struct Reference	1050
10.368.1.Detailed Description	1050
10.369.1ln::util::ilcell< T > Struct Template Reference	1050
10.369.1.Detailed Description	1051
10.370.1ln::util::line_graph< G > Class Template Reference	1051
10.370.1.Detailed Description	1052
10.370.2.Member Typedef Documentation	1053
10.370.2.1.ledge_fwd_iter	1053
10.370.2.2.edge_nbh_edge_fwd_iter	1053
10.370.2.3.edges_t	1053
10.370.2.4.vertex_nbh_edge_fwd_iter	1053
10.370.2.5.vertex_nbh_vertex_fwd_iter	1053
10.370.2.6.vertices_t	1053
10.370.3.Member Function Documentation	1053
10.370.3.1.1e_ith_nbh_edge	1053
10.370.3.2.2e_nmax	1054
10.370.3.3.edge	1054
10.370.3.4.graph	1054
10.370.3.5.has	1054
10.370.3.6.has	1054
10.370.3.7.has_e	1055
10.370.3.8.has_v	1055
10.370.3.9.is_subgraph_of	1055

10.370.3.101	1055
10.370.3.102	1055
10.370.3.102_ith_nbh_edge	1055
10.370.3.103_ith_nbh_vertex	1056
10.370.3.104_nmax	1056
10.370.3.105_vertex	1056
10.371.mln::util::nil Struct Reference	1056
10.371.Detailed Description	1057
10.372.mln::util::node< T, R > Class Template Reference	1057
10.372.Detailed Description	1057
10.373.mln::util::object_id< Tag, V > Class Template Reference	1057
10.373.Detailed Description	1058
10.373.Member Typedef Documentation	1059
10.373.2.lvalue_t	1059
10.373.Constructor & Destructor Documentation	1059
10.373.3.lobject_id	1059
10.374.mln::util::ord< T > Struct Template Reference	1059
10.374.Detailed Description	1059
10.375.mln::util::ord_pair< T > Struct Template Reference	1059
10.375.Detailed Description	1060
10.375.Member Function Documentation	1061
10.375.2.lchange_both	1061
10.375.2.2change_first	1061
10.375.2.3change_second	1061
10.375.2.4first	1061
10.375.2.5second	1061
10.376.mln::util::pix< I > Struct Template Reference	1062
10.376.Detailed Description	1062
10.376.Member Typedef Documentation	1062
10.376.2.lpsite	1062
10.376.2.2value	1062
10.376.Constructor & Destructor Documentation	1063
10.376.3.lpix	1063
10.376.Member Function Documentation	1063
10.376.4.lima	1063
10.376.4.2p	1063

10.376.4.3v	1063
10.377. util::set< T > Class Template Reference	1063
10.377.1. Detailed Description	1065
10.377.2. Member Typedef Documentation	1066
10.377.2.1. bkd_eter	1066
10.377.2.2. eter	1066
10.377.2.3. element	1066
10.377.2.4. fwd_eter	1066
10.377.3. Constructor & Destructor Documentation	1066
10.377.3.1. set	1066
10.377.4. Member Function Documentation	1066
10.377.4.1. clear	1066
10.377.4.2. first_element	1066
10.377.4.3. has	1067
10.377.4.4. insert	1067
10.377.4.5. insert	1067
10.377.4.6. is_empty	1068
10.377.4.7. last_element	1068
10.377.4.8. memory_size	1068
10.377.4.9. elements	1068
10.377.4.10. operator[]	1068
10.377.4.11. remove	1069
10.377.4.12. set_vector	1069
10.378. util::site_pair< P > Class Template Reference	1069
10.378.1. Detailed Description	1070
10.378.2. Member Function Documentation	1070
10.378.2.1. first	1070
10.378.2.2. pair	1071
10.378.2.3. second	1071
10.379. util::soft_heap< T, R > Class Template Reference	1071
10.379.1. Detailed Description	1072
10.379.2. Member Typedef Documentation	1072
10.379.2.1. element	1072
10.379.3. Constructor & Destructor Documentation	1072
10.379.3.1. soft_heap	1072
10.379.3.2. ~soft_heap	1073

10.379.4	Member Function Documentation	1073
10.379.4.1	clear	1073
10.379.4.2	is_empty	1073
10.379.4.3	is_valid	1073
10.379.4.4	elements	1073
10.379.4.5	pop_front	1073
10.379.4.6	push	1074
10.379.4.7	push	1074
10.380	std::timer Class Reference	1074
10.380	Detailed Description	1075
10.381	std::tracked_ptr< T > Struct Template Reference	1075
10.381	Detailed Description	1075
10.381	Constructor & Destructor Documentation	1076
10.381.2	tracked_ptr	1076
10.381.2	tracked_ptr	1076
10.381.2.3	~tracked_ptr	1076
10.381.3	Member Function Documentation	1076
10.381.3.1	operator bool	1076
10.381.3.2	operator!	1076
10.381.3.3	operator->	1076
10.381.3.4	operator->	1076
10.381.3.5	operator=	1077
10.381.3.6	operator=	1077
10.382	std::tree< T > Class Template Reference	1077
10.382	Detailed Description	1078
10.382	Constructor & Destructor Documentation	1078
10.382.2	tree	1078
10.382.2	tree	1078
10.382.3	Member Function Documentation	1078
10.382.3.1	ladd_tree_down	1078
10.382.3.2	add_tree_up	1078
10.382.3.3	check_consistency	1079
10.382.3.4	main_branch	1079
10.382.3.5	root	1079
10.383	std::tree_node< T > Class Template Reference	1079
10.383	Detailed Description	1080

10.383.2	Constructor & Destructor Documentation	1081
10.383.2.1	tree_node	1081
10.383.2.2	tree_node	1081
10.383.3	Member Function Documentation	1081
10.383.3.1	add_child	1081
10.383.3.2	add_child	1081
10.383.3.3	check_consistency	1082
10.383.3.4	children	1082
10.383.3.5	children	1082
10.383.3.6	delete_tree_node	1082
10.383.3.7	elt	1082
10.383.3.8	elt	1083
10.383.3.9	parent	1083
10.383.3.10	print	1083
10.383.3.11	search	1083
10.383.3.12	search_rec	1084
10.383.3.13	set_parent	1084
10.384.1	util::vertex< G > Class Template Reference	1084
10.384.2	Detailed Description	1086
10.384.3	Member Typedef Documentation	1086
10.384.3.1	Category	1086
10.384.3.2	graph_t	1086
10.384.3.3	id_t	1086
10.384.3.4	id_value_t	1086
10.384.4	Constructor & Destructor Documentation	1086
10.384.4.1	vertex	1086
10.384.5	Member Function Documentation	1087
10.384.5.1	change_graph	1087
10.384.5.2	edge_with	1087
10.384.5.3	graph	1087
10.384.5.4	id	1087
10.384.5.5	invalidate	1087
10.384.5.6	is_valid	1087
10.384.5.7	nth_nbh_edge	1088
10.384.5.8	nth_nbh_vertex	1088
10.384.5.9	nmax_nbh_edges	1088

10.384.4.10max_nbh_vertices	1088
10.384.4.11operator vertex_id_t	1088
10.384.4.12other	1088
10.384.4.13update_id	1088
10.385.mln::util::yes Struct Reference	1089
10.385.Detailed Description	1089
10.386.mln::Value< E > Struct Template Reference	1089
10.386.Detailed Description	1090
10.387.mln::value::float01 Class Reference	1091
10.387.Detailed Description	1092
10.387.Member Typedef Documentation	1092
10.387.2.lenc	1092
10.387.2.2equiv	1092
10.387.Constructor & Destructor Documentation	1092
10.387.3.lfloat01	1092
10.387.3.2float01	1092
10.387.3.3float01	1092
10.387.Member Function Documentation	1092
10.387.4.lnbits	1092
10.387.4.2operator float	1092
10.387.4.3set_nbits	1093
10.387.4.4to_nbits	1093
10.387.4.5value	1093
10.387.4.6value_ind	1093
10.388.mln::value::float01_f Struct Reference	1093
10.388.Detailed Description	1094
10.388.Constructor & Destructor Documentation	1094
10.388.2.lfloat01_f	1094
10.388.2.2float01_f	1094
10.388.Member Function Documentation	1094
10.388.3.loperator float	1094
10.388.3.2operator=	1094
10.388.3.3value	1094
10.389.mln::value::graylevel< n > Struct Template Reference	1094
10.389.Detailed Description	1096
10.389.Constructor & Destructor Documentation	1096

10.389.2.1graylevel	1096
10.389.2.2graylevel	1096
10.389.2.3graylevel	1096
10.389.2.4graylevel	1096
10.389.2.5graylevel	1097
10.389.3.Member Function Documentation	1097
10.389.3.1operator=	1097
10.389.3.2operator=	1097
10.389.3.3operator=	1097
10.389.3.4operator=	1097
10.389.3.5to_float	1097
10.389.3.6value	1097
10.390.mln::value::graylevel_f Struct Reference	1098
10.390.Detailed Description	1099
10.390.Constructor & Destructor Documentation	1099
10.390.2.1graylevel_f	1099
10.390.2.2graylevel_f	1099
10.390.2.3graylevel_f	1099
10.390.2.4graylevel_f	1099
10.390.2.5graylevel_f	1099
10.390.3.Member Function Documentation	1099
10.390.3.1operator graylevel< n >	1099
10.390.3.2operator=	1099
10.390.3.3operator=	1100
10.390.3.4operator=	1100
10.390.3.5operator=	1100
10.390.3.6value	1100
10.391.mln::value::int_s< n > Struct Template Reference	1100
10.391.Detailed Description	1102
10.391.Constructor & Destructor Documentation	1102
10.391.2.1int_s	1102
10.391.2.2int_s	1102
10.391.2.3int_s	1102
10.391.3.Member Function Documentation	1102
10.391.3.1operator int	1102
10.391.3.2operator=	1102

10.391.4	Member Data Documentation	1102
10.391.4.1	lone	1102
10.391.4.2	zero	1103
10.392	ln::value::int_u< n > Struct Template Reference	1103
10.392.1	Detailed Description	1104
10.392.2	Constructor & Destructor Documentation	1104
10.392.2.1	int_u	1104
10.392.2.2	int_u	1104
10.392.2.3	int_u	1104
10.392.3	Member Function Documentation	1104
10.392.3.1	next	1104
10.392.3.2	operator unsigned	1104
10.392.3.3	operator-	1105
10.392.3.4	operator=	1105
10.393	ln::value::int_u_sat< n > Struct Template Reference	1105
10.393.1	Detailed Description	1106
10.393.2	Constructor & Destructor Documentation	1106
10.393.2.1	int_u_sat	1106
10.393.2.2	int_u_sat	1106
10.393.3	Member Function Documentation	1106
10.393.3.1	operator int	1106
10.393.3.2	operator+=	1107
10.393.3.3	operator-=	1107
10.393.3.4	operator=	1107
10.393.4	Member Data Documentation	1107
10.393.4.1	lone	1107
10.393.4.2	zero	1107
10.394	ln::value::Integer< E > Struct Template Reference	1107
10.394.1	Detailed Description	1108
10.395	ln::value::Integer< void > Struct Template Reference	1108
10.395.1	Detailed Description	1108
10.396	ln::value::label< n > Struct Template Reference	1108
10.396.1	Detailed Description	1109
10.396.2	Member Typedef Documentation	1109
10.396.2.1	lenc	1109
10.396.3	Constructor & Destructor Documentation	1110

10.396.3.llabel	1110
10.396.3.2label	1110
10.396.3.3label	1110
10.396.4.Member Function Documentation	1110
10.396.4.lnext	1110
10.396.4.2operator unsigned	1110
10.396.4.3operator++	1110
10.396.4.4operator--	1110
10.396.4.5operator=	1111
10.396.4.6operator=	1111
10.396.4.7prev	1111
10.397.ln::value::lut_vec< S, T > Struct Template Reference	1111
10.397.Detailed Description	1112
10.397.2.Member Typedef Documentation	1112
10.397.2.lbk_d_viter	1112
10.397.2.2fwd_viter	1112
10.397.2.3value	1112
10.397.3.Constructor & Destructor Documentation	1113
10.397.3.llut_vec	1113
10.397.3.2lut_vec	1113
10.397.3.3lut_vec	1113
10.397.4.Member Function Documentation	1113
10.397.4.lhas	1113
10.397.4.2index_of	1113
10.397.4.3nvalues	1113
10.397.4.4operator[]	1114
10.398.ln::value::proxy< I > Class Template Reference	1114
10.398.Detailed Description	1115
10.398.2.Member Typedef Documentation	1115
10.398.2.lenc	1115
10.398.2.2equiv	1115
10.398.3.Constructor & Destructor Documentation	1115
10.398.3.lproxy	1115
10.398.3.2proxy	1116
10.398.3.3~proxy	1116
10.398.4.Member Function Documentation	1116

10.398.4.1operator=	1116
10.398.4.2operator=	1116
10.398.4.3to_value	1116
10.399.1Inn::value::qt::rgb32 Struct Reference	1116
10.399.1.Detailed Description	1117
10.399.1.Constructor & Destructor Documentation	1117
10.399.2.1rgb32	1117
10.399.2.2rgb32	1117
10.399.2.3rgb32	1117
10.399.2.4rgb32	1118
10.399.3.Member Function Documentation	1118
10.399.3.1operator=	1118
10.399.3.2red	1118
10.399.4.Member Data Documentation	1118
10.399.4.1zero	1118
10.400.1Inn::value::rgb< n > Struct Template Reference	1118
10.400.1.Detailed Description	1119
10.400.1.Constructor & Destructor Documentation	1119
10.400.2.1rgb	1119
10.400.2.2rgb	1119
10.400.2.3rgb	1119
10.400.2.4rgb	1119
10.400.3.Member Function Documentation	1120
10.400.3.1operator=	1120
10.400.3.2red	1120
10.400.4.Member Data Documentation	1120
10.400.4.1zero	1120
10.401.1Inn::value::set< T > Struct Template Reference	1120
10.401.1.Detailed Description	1120
10.401.2.Member Function Documentation	1121
10.401.2.1the	1121
10.402.1Inn::value::sign Class Reference	1121
10.402.1.Detailed Description	1122
10.402.2.Member Typedef Documentation	1122
10.402.2.1enc	1122
10.402.2.2equiv	1122

10.402.	Constructor & Destructor Documentation	1122
10.402.3.	1sign	1122
10.402.3.	2sign	1122
10.402.3.	3sign	1122
10.402.	Member Function Documentation	1123
10.402.4.	1operator int	1123
10.402.4.	2operator=	1123
10.402.	Member Data Documentation	1123
10.402.5.	1one	1123
10.402.5.	2zero	1123
10.402.	1n::value::stack_image< n, I > Struct Template Reference	1123
10.403.	Detailed Description	1124
10.403.	Member Typedef Documentation	1124
10.403.2.	1domain_t	1124
10.403.2.	2value	1124
10.403.2.	3psite	1125
10.403.2.	4value	1125
10.403.2.	5skeleton	1125
10.403.2.	6value	1125
10.403.	Constructor & Destructor Documentation	1125
10.403.3.	1stack_image	1125
10.403.	Member Function Documentation	1125
10.403.4.	1is_valid	1125
10.403.4.	2operator()	1125
10.403.4.	3operator()	1126
10.404.	1n::value::super_value< sign > Struct Template Reference	1126
10.404.	Detailed Description	1126
10.405.	1n::value::value_array< T, V > Struct Template Reference	1126
10.405.	Detailed Description	1126
10.405.	Constructor & Destructor Documentation	1127
10.405.2.	1value_array	1127
10.405.	Member Function Documentation	1127
10.405.3.	1operator()	1127
10.405.3.	2operator[]	1127
10.405.3.	3vset	1127
10.406.	1n::Vertex< E > Struct Template Reference	1127

10.406. Detailed Description	1127
10.407. <code>ln::vertex_image< P, V, G ></code> Class Template Reference	1128
10.407. Detailed Description	1128
10.407. Member Typedef Documentation	1129
10.407.2. <code>lgraph_t</code>	1129
10.407.2. <code>nbh_t</code>	1129
10.407.2. <code>site_function_t</code>	1129
10.407.2. <code>skeleton</code>	1129
10.407.2. <code>win_t</code>	1129
10.407. Constructor & Destructor Documentation	1129
10.407.3. <code>lvertex_image</code>	1129
10.407. Member Function Documentation	1130
10.407.4. <code>operator()</code>	1130
10.408. <code>ln::violent_cast_image< T, I ></code> Struct Template Reference	1130
10.408. Detailed Description	1130
10.408. Member Typedef Documentation	1131
10.408.2. <code>lvalue</code>	1131
10.408.2. <code>rvalue</code>	1131
10.408.2. <code>skeleton</code>	1131
10.408.2. <code>value</code>	1131
10.408. Constructor & Destructor Documentation	1131
10.408.3. <code>lvioleat_image</code>	1131
10.408. Member Function Documentation	1131
10.408.4. <code>operator()</code>	1131
10.408.4. <code>operator()</code>	1131
10.409. <code>ln::w_window< D, W ></code> Struct Template Reference	1132
10.409. Detailed Description	1133
10.409. Member Typedef Documentation	1133
10.409.2. <code>lbgd_qiter</code>	1133
10.409.2. <code>dpstte</code>	1133
10.409.2. <code>fwq_qiter</code>	1133
10.409.2. <code>weight</code>	1133
10.409. Constructor & Destructor Documentation	1134
10.409.3. <code>lw_window</code>	1134
10.409. Member Function Documentation	1134
10.409.4. <code>lclear</code>	1134

10.409.4.2insert	1134
10.409.4.3is_symmetric	1134
10.409.4.4std_vector	1134
10.409.4.5sym	1134
10.409.4.6w	1135
10.409.4.7weights	1135
10.409.4.8win	1135
10.409.5.Friends And Related Function Documentation	1135
10.409.5.1operator<<	1135
10.409.5.2operator==	1135
10.410.mln::Weighted_Window< E > Struct Template Reference	1135
10.410.1.Detailed Description	1136
10.410.2.Friends And Related Function Documentation	1136
10.410.2.1operator-	1136
10.411.mln::win::backdiag2d Struct Reference	1137
10.411.1.Detailed Description	1137
10.411.2.Constructor & Destructor Documentation	1137
10.411.2.1backdiag2d	1137
10.411.3.Member Function Documentation	1138
10.411.3.1length	1138
10.412.mln::win::ball< G, C > Struct Template Reference	1138
10.412.1.Detailed Description	1138
10.412.2.Constructor & Destructor Documentation	1138
10.412.2.1ball	1138
10.412.3.Member Function Documentation	1139
10.412.3.1diameter	1139
10.413.mln::win::cube3d Struct Reference	1139
10.413.1.Detailed Description	1139
10.413.2.Constructor & Destructor Documentation	1139
10.413.2.1cube3d	1139
10.413.3.Member Function Documentation	1140
10.413.3.1length	1140
10.414.mln::win::cuboid3d Struct Reference	1140
10.414.1.Detailed Description	1140
10.414.2.Constructor & Destructor Documentation	1141
10.414.2.1cuboid3d	1141

10.414.3	Member Function Documentation	1141
10.414.3.1	depth	1141
10.414.3.2	height	1141
10.414.3.3	volume	1142
10.414.3.4	width	1142
10.415	ln::win::diag2d Struct Reference	1142
10.415.1	Detailed Description	1142
10.415.2	Constructor & Destructor Documentation	1142
10.415.2.1	diag2d	1142
10.415.3	Member Function Documentation	1143
10.415.3.1	length	1143
10.416	ln::win::line< M, i, C > Struct Template Reference	1143
10.416.1	Detailed Description	1143
10.416.2	Member Enumeration Documentation	1144
10.416.2.1	"@87	1144
10.416.3	Constructor & Destructor Documentation	1144
10.416.3.1	line	1144
10.416.4	Member Function Documentation	1144
10.416.4.1	length	1144
10.416.4.2	size	1144
10.417	ln::win::multiple< W, F > Class Template Reference	1145
10.417.1	Detailed Description	1145
10.418	ln::win::multiple_size< n, W, F > Class Template Reference	1145
10.418.1	Detailed Description	1145
10.419	ln::win::octagon2d Struct Reference	1145
10.419.1	Detailed Description	1146
10.419.2	Constructor & Destructor Documentation	1146
10.419.2.1	octagon2d	1146
10.419.3	Member Function Documentation	1146
10.419.3.1	larea	1146
10.419.3.2	length	1146
10.420	ln::win::rectangle2d Struct Reference	1147
10.420.1	Detailed Description	1147
10.420.2	Constructor & Destructor Documentation	1147
10.420.2.1	rectangle2d	1147
10.420.3	Member Function Documentation	1148

10.420.3.larea	1148
10.420.3.2height	1148
10.420.3.3std_vector	1148
10.420.3.4width	1148
10.421.mn::Window< E > Struct Template Reference	1148
10.421.Detailed Description	1149
10.422.mn::window< D > Class Template Reference	1149
10.422.Detailed Description	1151
10.422.2Member Typedef Documentation	1151
10.422.2.1bkd_qiter	1151
10.422.2.2fwd_qiter	1151
10.422.2.3qiter	1151
10.422.2.4regular	1151
10.422.3Constructor & Destructor Documentation	1151
10.422.3.lwindow	1151
10.422.4Member Function Documentation	1152
10.422.4.1clear	1152
10.422.4.2delta	1152
10.422.4.3dp	1152
10.422.4.4has	1152
10.422.4.5insert	1152
10.422.4.6insert	1152
10.422.4.7insert	1153
10.422.4.8s_centered	1153
10.422.4.9s_empty	1153
10.422.4.10s_symmetric	1153
10.422.4.11print	1153
10.422.4.12ze	1153
10.422.4.13d_vector	1154
10.422.4.14ym	1154
10.422.5Friends And Related Function Documentation	1154
10.422.5.operator==	1154
10.423.mn::world::inter_pixel::is_separator Struct Reference	1154
10.423.Detailed Description	1155
10.424.trait::graph< I > Struct Template Reference	1155
10.424.Detailed Description	1156

10.425. frait::graph< mln::complex_image< 1, G, V > > Struct Template Reference	1156
10.425. Detailed Description	1156
10.426. frait::graph< mln::image2d< T > > Struct Template Reference	1156
10.426. Detailed Description	1156

Chapter 1

Documentation of milena

1.1 Introduction

This is the documentation of Milena.

1.2 Overview of Milena.

- [mln](#)
- [mln::accu](#)
- [mln::algebra](#)
- [mln::arith](#)
- [mln::binarization](#)
- [mln::border](#)
- [mln::canvas](#)
- [mln::convert](#)
- [mln::data](#)
- [mln::debug](#)
- [mln::display](#)
- [mln::draw](#)
- [mln::estim](#)
- [mln::extension](#)
- [mln::fun](#)
- [mln::geom](#)
- [mln::graph](#)
- [mln::histo](#)

- [mln::io](#)
- [mln::labeling](#)
- [mln::data](#)
- [mln::linear](#)
- [mln::literal](#)
- [mln::logical](#)
- [mln::make](#)
- [mln::math](#)
- [mln::metal](#)
- [mln::morpho](#)
- [mln::norm](#)
- [mln::opt](#)
- [mln::pw](#)
- [mln::registration](#)
- [mln::set](#)
- [mln::tag](#)
- [mln::test](#)
- [mln::topo](#)
- [mln::trace](#)
- [mln::trait](#)
- [mln::transform](#)
- [mln::util](#)
- [mln::value](#)
- [mln::win](#)

1.3 Copyright and License.

Copyright (C) 2007, 2008, 2009, 2010, 2011 EPITA Research and Development (LRDE)

This documentation is part of Olena.

Olena is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2 of the License.

Olena is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Olena. If not, see <http://www.gnu.org/licenses/>.

Chapter 2

Quick Reference Guide

- [Installation](#)
- [Foreword](#)
- [Site](#)
- [Site set](#)
- [Image](#)
- [Structural elements: Window and neighborhood](#)
- [Sites, psites and dpoints](#)
- [Iterators](#)
- [Memory management](#)
- [Basic routines](#)
- [Input / Output](#)
- [Graphs and images](#)
- [Useful global variables](#)
- [Useful macros](#)
- [Common Compilation Errors](#)

2.1 Installation

2.1.1 Requirements

2.1.1.1 To compile the user examples

2.1.1.2 To compile the documentation (Optional)

2.1.1.3 To develop in Olena

2.1.2 Getting Olena

2.1.3 Building Olena

2.2 Foreword

2.2.1 Generality

2.2.2 Directory hierarchy

2.2.3 Writing and compiling a program with Olena

2.3 Site

2.4 Site set

Iterators

2.4.1 Basic interface

2.4.2 Optional interface

```
box2d b(2,3);

// The bbox can be retrived in constant time.
std::cout << b.bbox() << std::endl;

// nsites can be retrieved in constant time.
std::cout << "nsites = " << b.nsites() << std::endl;

[(0,0)..(1,2)]
nsites = 6

p_array<point2d> arr;
arr.insert(point2d(1,0));
arr.insert(point2d(1,1));

// The bbox is computed thanks to bbox() algorithm.
box2d box = geom::bbox(arr);
std::cout << box << std::endl;

// p_array provides nsites(),
```

```
// it can be retrieved in constant time.
std::cout << "nsites = " << arr.nsites() << std::endl;

[(1,0)..(1,1)]
nsites = 2
```

2.5 Image

2.5.1 Definition

2.5.2 Possible image types

2.5.3 Possible value types

2.5.4 Domain

```
// Define a box2d from (-2,-3) to (3,5).
box2d b = make::box2d(-2,-3, 3,5);
// Initialize an image with b as domain.
image2d<int> ima(b);

std::cout << "b = " << b << std::endl;
std::cout << "domain = " << ima.domain() << std::endl;

b = [(-2,-3)..(3,5)]
domain = [(-2,-3)..(3,5)]

// Create an image on a 2D box
// with 10 columns and 10 rows.
image2d<bool> ima(make::box2d(10, 10));

m1n_site_(image2d<bool>) p1(20, 20);
m1n_site_(image2d<bool>) p2(3, 3);

std::cout << "has(p1)? "
          << (ima.has(p1) ? "true" : "false")
          << std::endl;

std::cout << "has(p2)? "
          << (ima.has(p2) ? "true" : "false")
          << std::endl;

has(p1)? false
has(p2)? true

point2d p(9,9);

// At (9, 9), both values change.
ima1(p) = 'M';
ima2(p) = 'W';

bool b = (ima1(p) == ima2(p));
std::cout << (b ? "True" : "False") << std::endl;

False
```

2.5.5 Border and extension

2.5.5.1 Image border

```

bool vals[3][3] = { { 0, 1, 1 },
                    { 1, 0, 0 },
                    { 1, 1, 0 } };

image2d<bool> ima_def = make::image(vals);
border::fill(ima_def, false);
debug::println_with_border(ima_def);

std::cout << "======" << std::endl << std::endl;

border::thickness = 0;
image2d<bool> ima_bt0 = make::image(vals);
debug::println_with_border(ima_bt0);

- - - - -
- - - - -
- - - - -
- - - | | - -
- - | - - - -
- - | | - - -
- - - - - - -
- - - - - - -
- - - - - - -

=====

- | |
| - -
| | -

```

2.5.5.2 Generality on image extension

imamorphed

2.5.5.3 Different extensions

```

image2d<rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");
box2d bbox_enlarged = lena.domain();
bbox_enlarged.enlarge(border::thickness);
mln_VAR(ima_roi, lena | fun::p2b::big_chess<box2d>(lena.domain(), 10));

```

2.5.5.3.1 Extension with a value

```

mln_VAR(ext_with_val, extended_to(extend(ima_roi, literal::blue), bbox_enlarged
));

```

2.5.5.3.2 Extension with a function

```

namespace mln
{

    struct my_ext : public Function_v2v<my_ext>
    {

```

```

typedef value::rgb8 result;

value::rgb8 operator()(const point2d& p) const
{
    if ((p.row() + p.col()) % 20)
        return literal::black;
    return literal::white;
}

};

} // end of namespace mln

mln_VAR(ext_with_fun, extended_to(extend(ima_roi, my_ext()), bbox_enlarged));

```

2.5.5.3.3 Extension with an image

```

mln_VAR(ext_with_ima, extend(ima_roi, lena));

// Default border size is set to 0.

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima1(2, 3);

std::cout << "ima1.has(0, 0) : "
            << ima1.has(point2d(0, 0)) << std::endl;

std::cout << "ima1.has(-3, 0) : "
            << ima1.has(point2d(-3, 0)) << std::endl;

std::cout << "ima1.has(2, 5) : "
            << ima1.has(point2d(2, 5)) << std::endl;

std::cout << "======" << std::endl;

// Set default border size to 0.
border::thickness = 0;

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima2(2, 3);

std::cout << "ima2.has(0, 0) : "
            << ima2.has(point2d(0, 0)) << std::endl;

std::cout << "ima2.has(-3, 0) : "
            << ima2.has(point2d(-3, 0)) << std::endl;

std::cout << "ima2.has(2, 5) : "
            << ima2.has(point2d(2, 5)) << std::endl;

ima1.has(0, 0) : 1
ima1.has(-3, 0) : 1
ima1.has(2, 5) : 1
=====
ima2.has(0, 0) : 1
ima2.has(-3, 0) : 0
ima2.has(2, 5) : 0

border::thickness = 30;

// Declare the image to be rotated.

```

```

image2d<value::rgb8> ima1(220, 220);
data::fill(ima1, literal::cyan);
border::fill(ima1, literal::yellow);
// Set an infinite extension.
mln_VAR(ima1, extend(ima1, pw::cst(literal::yellow)));

// Declare the output image.
image2d<value::rgb8> ima2(220, 220);
data::fill(ima2, literal::cyan);
border::fill(ima2, literal::yellow);

box2d extended_domain= ima1.domain();
extended_domain.enlarge(border::thickness);

// Draw the domain bounding box
draw::box(ima1, geom::bbox(ima1), literal::red);
// Save the image, including its border.
doc::ppmsave(ima1 | extended_domain, "ima2d-rot");

// Define and apply a point-wise rotation
fun::x2x::rotation<2,float> rot1(0.5, literal::zero);
image2d<value::rgb8>::fwd_piter p(ima1.domain());
for_all(p)
{
    algebra::vec<2,float> pv = p.to_site().to_vec();
    algebra::vec<2,float> v = rot1.inv()(pv);
    ima2(p) = ima1(v);
}

draw::box(ima2, ima2.bbox(), literal::red);
doc::ppmsave(extended_to(ima2, extended_domain), "ima2d-rot");

my_routine(ima | ima.domain());

```

2.5.6 Interface

2.5.7 Load and save images

```

image2d<bool> ima;
io::pbn::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbn::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");

```

2.5.8 Create an image

```

// Build an empty image;
image2d<value::int_u8> img1a;

// Build an image with 2 rows
// and 3 columns sites
image2d<value::int_u8> img1b(box2d(2, 3));
image2d<value::int_u8> img1c(2, 3);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

```



```

image2d<value::int_u8> img2a(2, 3);
image2d<value::int_u8> img2b;

initialize(img2b, img2a);
data::fill(img2b, img2a);

```

Fill

2.5.9 Access and modify values

```

box2d b(2,3);
image2d<value::int_u8> ima(b);

// On image2d, Site <=> point2d
point2d p(1, 2);

// Associate '9' as value for the site/point2d (1,2).
// The value is returned by reference and can be changed.
opt::at(ima, 1,2) = 9;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

std::cout << "---" << std::endl;

// Associate '2' as value for the site/point2d (1,2).
// The value is returned by reference
// and can be changed as well.
ima(p) = 2;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

opt::at(ima, 1,2) = 9
ima(p) = 9
---
opt::at(ima, 1,2) = 2
ima(p) = 2

```

Iterators

2.5.10 Image size

```

image2d<int> ima(make::box2d(0,0, 10,12));

std::cout << "nrows = " << ima.nrows()
          << " - "
          << "ncols = " << ima.ncols()
          << std::endl;

nrows = 11 - ncols = 13

```

2.6 Structural elements: Window and neighborhood

2.6.1 Define an element

2.6.1.1 Window

2.6.1.2 Neighborhood

```
label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);
```

2.6.1.3 Custom structural elements

```
window2d win;
win.insert(-1, -1);
win.insert(-1, 0);
win.insert(-1, 1);
```

```
o -
o X
o -
```

```
bool b[9] = { 1, 0, 0,
              1, 0, 0,
              1, 0, 0 };
```

```
bool b2[3][3] = { { 1, 0, 0 },
                  { 1, 0, 0 },
                  { 1, 0, 0 } };
```

```
window2d win = convert::to<window2d>(b);
window2d win2 = convert::to<window2d>(b2);
```

2.6.1.4 Conversion between Neighborhoods and Windows

2.7 Sites, psites and dpoints

2.7.1 Need for site

```
c 0 1 2 3
r
+---+---+
0 | |x| | |
+---+---+
1 | | | | |
+---+---+
```

2.7.2 Need for psite

```
unsigned my_values(const mln::point2d& p)
{
    if (p.row() == 0)
        return 8;
    return 9;
}
```

```

p_array<point2d> arr;
arr.append(point2d(3, 6));
arr.append(point2d(3, 7));
arr.append(point2d(3, 8));
arr.append(point2d(4, 8));
arr.append(point2d(4, 9));

mln_VAR(ima, my_values | arr);

c  6 7 8 9
r
  +--+--+
3  | |x| |
  +--+--+--+
4      | | |
      +--+--+

arr[] = 0 1 2 3 4
      +--+--+--+--+
      | |x| | | |
      +--+--+--+--+

```

2.7.3 From psite to site

2.7.4 Dpoint

```

dpoint2d dp(-1,0);
point2d p(1,1);

std::cout << p + dp << std::endl;

(0,1)

```

2.8 Iterators

```

box2d b(3, 2);
mln_piter_(box2d) p(b);

for_all(p)
  std::cout << p; //prints every site coordinates.

(0,0) (0,1) (1,0) (1,1) (2,0) (2,1)

template <typename I>
void fill(I& ima, mln_value(I) v)
{
  mln_piter(I) p(ima.domain());
  for_all(p)
    ima(p) = v;
}

template <typename I, typename J>
void paste(const I& data, J& dest)
{
  mln_piter(I) p(data.domain());
  for_all(p)
    dest(p) = data(p);
}

```

Useful macros

2.9 Memory management

```

image2d<int> ima1(box2d(2, 3));
image2d<int> ima2;
point2d p(1,2);

ima2 = ima1; // ima1.id() == ima2.id()
// and both point to the same memory area.

ima2(p) = 2; // ima1 is modified as well.

// prints "2 - 2"
std::cout << ima2(p) << " - " << ima1(p) << std::endl;
// prints "true"
std::cout << (ima2.id_() == ima1.id_()) << std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima3 = duplicate(ima1); // Makes a deep copy.

point2d p(2, 2);
ima3(p) = 3;

std::cout << ima3(p) << " - " << ima1(p) << std::endl;
std::cout << (ima3.id_() == ima1.id_()) << std::endl;

3 - 0
0

```

2.10 Basic routines

2.10.1 Fill

```

image2d<char> imga(5, 5);

data::fill(imga, 'a');

data::fill((imga | box2d(1,2)).rw(), 'a');

```

2.10.2 Paste

```

image2d<unsigned char> imgb(make::box2d(5,5, 7,8));
// Initialize imga with the same domain as imgb.
image2d<unsigned char> imga(imgb.domain());

// Initialize the image values.
data::fill(imgb, 'b');

// Paste the content of imgb in imga.
data::paste(imgb, imga);

debug::println(imga);

98 98 98 98
98 98 98 98
98 98 98 98

image2d<int> ima1(5, 5);

```

```

image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
           << std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
           << std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
           << std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
           << std::endl;

```

2.10.3 Blobs

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

```

2.10.4 Logical not

```

bool vals[5][5] = {
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0}
};
image2d<bool> ima = make::image(vals);

image2d<bool> ima_neg = logical::not_(ima);

logical::not_inplace(ima);

```

2.10.5 Compute

2.10.5.1 Accumulators

```

data::compute(accu::meta::stat::max, ima);

data::compute(accu::meta::stat::max(), ima);

```

2.10.5.2 Example with labeling::compute()

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},

```

```

        {0, 1, 1, 0, 0},
        {0, 0, 0, 0, 0},
        {1, 1, 0, 1, 0},
        {1, 0, 1, 1, 1},
        {1, 0, 0, 0, 0}
    };
    image2d<bool> ima = make::image(vals);

    label_8 nlabels;
    image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

    util::array<box2d> boxes =
        labeling::compute(accu::meta::shape::bbox(),
                        lbl,
                        nlabels);

    for (unsigned i = 1; i <= nlabels; ++i)
        std::cout << boxes[i] << std::endl;

    [(0,1)..(1,2)]
    [(3,0)..(5,1)]
    [(3,2)..(4,4)]

    unsigned nsites = geom::nsites(ima);

```

2.10.6 Working with parts of an image

```

//function_p2b
bool my_function_p2b(mln::point2d p);

//function_p2v
//V is the value type used in the image.
template <typename V>
V my_function_p2v(mln::point2d p);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
    image2d<bool> ima = make::image(vals);

```

2.10.6.1 Restrict an image with a site set

```

p_array<point2d> arr;

// We add two points in the array.
arr.append(point2d(0, 1));
arr.append(point2d(4, 0));

// We restrict the image to the sites
// contained in arr and fill these ones
// with 0.
// We must call "rw()" here.
data::fill((ima | arr).rw(), 0);

debug::println((ima | arr));

```

```
mln_VAR(ima2, ima | arr);
// We do not need to call "rw()" here.
data::fill(ima2, 0);
```

```
-
```

```
-
```

```
-
```

```
-
```

2.10.6.2 Restrict an image with a predicate

```
label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

mln_VAR(lbl_2, lbl | (pw::value(lbl) == pw::cst(2u)));

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | lbl_2.domain()).rw(), literal::red);

label_8 nlabels;
image2d<label_8> lab = labeling::blobs(ima, c4(), nlabels);

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | (pw::value(lab) == pw::cst(2u))).rw(), literal::red);
```

2.10.6.3 Restrict an image with a C function

```
bool row_oddity(mln::point2d p)
{
    return p.row() % 2;
}

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | row_oddity).rw(), literal::red);

ima | sub_D

0 1 0
1 1 1

mln_VAR(imab1, ima | (pw::value(ima) == pw::cst(1u)));
```

```

1
1 1 1

box2d b1(1,0, 1, 2);
mln_VAR(imac, imab1 | b1);

// Print:
// 1 1 1
debug::println(imac);

box2d b2(0,0, 1, 1);
// Will fail at runtime.
// ima.domain().has((0,0)) is false.
mln_VAR(imad, imab1 | b2);
debug::println(imad);

ima / sub_D

```

2.11 Input / Output

2.11.1 ImageMagick

2.11.2 GDCM

2.12 Graphs and images

2.12.1 Description

2.12.2 Example

```

      0 1 2 3 4
    .-----
0 | 0           2
1 |   \       / |
2 |     1     |
3 |       \   |
4 |         3-4

util::graph g;

for (unsigned i = 0; i < 5; ++i)
    g.add_vertex(); // Add vertex 'i';

g.add_edge(0, 1); // Associated to edge 0.
g.add_edge(1, 2); // Associated to edge 1.
g.add_edge(1, 3); // Associated to edge 2.
g.add_edge(3, 4); // Associated to edge 3.
g.add_edge(4, 2); // Associated to edge 4.

typedef fun::i2v::array<point2d> F;
F f(5); // We need to map 5 vertices.
f(0) = point2d(0, 0);
f(1) = point2d(2, 2);
f(2) = point2d(0, 4);
f(3) = point2d(4, 3);
f(4) = point2d(4, 4);

```



```

typedef p_vertices<util::graph, F> pv_t;
pv_t pv(g, f);

template <typename S>
struct viota_t : public mln::Function_v2v< viota_t<S> >
{
    typedef unsigned result;

    viota_t(unsigned size)
    {
        v_.resize(size);
        for(unsigned i = 0; i < size; ++i)
            v_[i] = 10 + i;
    }

    unsigned
    operator()(const mln_psite(S)& p) const
    {
        return v_[p.v().id()];
    }

protected:
    std::vector<result> v_;
};

// Constructs an image
viota_t<pv_t> viota(pv.nsites());
mln_VAR(graph_vertices_ima, viota | pv);

//Prints each vertex and its associated data.
mln_piter_(graph_vertices_ima_t) p(graph_vertices_ima.domain());
for_all(p)
    std::cout << "graph_vertices_ima(" << p << ") = "
               << graph_vertices_ima(p) << std::endl;

graph_vertices_ima((0,0)) = 10
graph_vertices_ima((2,2)) = 11
graph_vertices_ima((0,4)) = 12
graph_vertices_ima((4,3)) = 13
graph_vertices_ima((4,4)) = 14

// Function which maps sites to data.
viota_t viota(g.v_nmax());

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Prints each vertex and its associated value.
for_all(v)
    std::cout << v << " : " << viota(v) << std::endl;

0 : 10
1 : 11
2 : 12
3 : 13
4 : 14

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Iterator on v's edges.
mln_vertex_nbh_edge_iter_(util::graph) e(v);

```

```

// Prints the graph
// List all edges for each vertex.
for_all(v)
{
    std::cout << v << " : ";
    for_all(e)
        std::cout << e << " ";
    std::cout << std::endl;
}

0 : (0,1)
1 : (0,1) (1,2) (1,3)
2 : (1,2) (2,4)
3 : (1,3) (3,4)
4 : (3,4) (2,4)

// Iterator on edges.
mIn_edge_iter_(util::graph) e(g);

// Iterator on edges adjacent to e.
mIn_edge_nbh_edge_iter_(util::graph) ne(e);

// Prints the graph
// List all adjacent edges for each edge.
for_all(e)
{
    std::cout << e << " : ";
    for_all(ne)
        std::cout << ne << " ";
    std::cout << std::endl;
}

(0,1) : (1,2) (1,3)
(1,2) : (0,1) (1,3) (2,4)
(1,3) : (0,1) (1,2) (3,4)
(3,4) : (1,3) (2,4)
(2,4) : (1,2) (3,4)

// Iterator on vertices.
mIn_vertex_iter_(util::graph) v(g);

// Iterator on vertices adjacent to v.
mIn_vertex_nbh_vertex_iter_(util::graph) nv(v);

// Prints the graph
// List all adjacent edges for each edge.
for_all(v)
{
    std::cout << v << " : ";
    for_all(nv)
        std::cout << nv << " ";
    std::cout << std::endl;
}

0 : 1
1 : 0 2 3
2 : 1 4
3 : 1 4
4 : 3 2

```

2.13 Useful global variables

2.14 Useful macros

2.14.1 Variable declaration macros

2.14.2 Iterator type macros

2.14.2.1 Default iterator types

2.14.2.2 Forward iterator types

2.14.2.3 Backward iterators

2.14.2.4 Graph iterators

2.15 Common Compilation Errors

Chapter 3

Tutorial

- [Welcome](#)
- [Installation](#)
- [Getting started with Milena](#)
- [Data representation](#)
- [Load and save images](#)
- [Create your first image](#)
- [Read and write images](#)
- [Regions of interest](#)

3.1 Welcome

3.1.1 How to learn Milena

3.1.2 Obtaining the library

3.1.3 Downloading the library

3.1.3.1 Downloading from SVN

[Installation](#) [Directory structure](#) [Join the mailing lists](#)

3.1.3.2 Downloading packaged releases

[Installation](#) [Directory structure](#) [Join the mailing lists](#) [Documentation](#)

3.1.4 Join the mailing lists

3.1.5 Directory structure

3.1.6 Documentation

3.1.7 Community and Support

[Join the mailing lists](#) [Contacts](#)

3.1.8 Project status

[Join the mailing lists](#)

3.1.9 A brief history of Milena

3.1.10 Contacts

[Installation](#)

3.2 Installation

3.2.1 Bootstrap (SVN Sources)

[Configure](#)

3.2.2 Configure

3.2.3 Install

[Optional compilation](#) [Installation content](#)

3.2.4 Optional compilation

3.2.4.1 Examples

3.2.4.2 Tools

3.2.4.3 Tests

3.2.5 Installation content

[Welcome](#) [Getting started with Milena](#)

3.3 Getting started with Milena

3.3.1 Getting familiar with genericity

```
// Java or C -like code

void fill(image *ima, unsigned char v)
{
    for (int i = 0; i < ima->nrows; ++i)
        for (int j = 0; j < ima->ncols; ++j)
            ima->data[i][j] = v;
}

template <typename I>
void fill(I& ima, mln_value(I) v)
{
    mln_piter(I) p(ima.domain());
    for_all(p)
        ima(p) = v;
}

fill(ima, literal::green);

box2d b(20,20);
fill((ima | b).rw(), literal::green);
```

3.3.2 First generic algorithm

```
#include <mln/core/image/image2d.hh>
#include <mln/core/image/dmorph/image_if.hh>
#include <mln/core/alias/neighb2d.hh>

#include <mln/data/fill.hh>

#include <mln/labeling/blobs.hh>
#include <mln/labeling/compute.hh>
#include <mln/labeling/blobs.hh>

#include <mln/data/compare.hh>

#include <mln/util/array.hh>

#include <mln/value/label_8.hh>

#include <mln/accu/math/count.hh>

#include <mln/pw/all.hh>

#include <tests/data.hh>
#include <doc/tools/sample_utils.hh>

namespace mln
{
    template <typename I, typename N>
    mln_concrete(I)
    my_algorithm(const Image<I>& ima_,
                 const Neighborhood<N>& nbh_)
    {
        trace::entering("my_algorithm");

        const I& ima = exact(ima_);
        const N& nbh = exact(nbh_);
```

```

mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;
}

} // end of namespace mln

template <typename I, typename N>
mln_concrete(I)
my_algorithm(const Image<I>& ima_,
             const Neighborhood<N>& nbh_)

    trace::entering("my_algorithm");

```

Debug hints

```

const I& ima = exact(ima_);
const N& nbh = exact(nbh_);
mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;

```


3.3.3 Compilation

3.3.3.1 Include path

3.3.3.2 Library linking

Input / Output

3.3.3.3 Disable Debug

3.3.3.4 Compiler optimization flags

3.3.3.4.1 GCC

3.3.3.4.2 Other compilers

3.3.4 Debug hints

3.3.4.1 Using assertions and GDB

3.3.4.2 Traces

```
// ...
trace::quiet = false;

labeling::blobs(ima, c4(), nlabels);

trace::quiet = true;

geom::bbox(ima);
// ...
```

3.3.4.3 Debug routines

```
image2d<int_u8> ima(5,5);
data::fill(ima, 2);
debug::println(ima);

2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2

image2d<int_u8> ima(5,5);
data::fill(ima, 2);
border::fill(ima, 7);
debug::println_with_border(ima);

7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 2 2 2 2 2 7 7 7
7 7 7 2 2 2 2 2 7 7 7
7 7 7 2 2 2 2 2 7 7 7
```

```

7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7

```

```

int_u8 vals[25] = { 100, 100, 200, 200, 230,
                   100, 100, 200, 230, 230,
                   140, 140, 140,  0,  0,
                   65, 186, 65, 127, 127,
                   65,  65, 65, 127, 127 };

image2d<int_u8> ima = make::image2d(vals);
image2d<rgb8> ima_color = labeling::colorize(rgb8(), ima, 230);

```

Installation Data representation

3.4 Data representation

3.4.1 Sites

```

point2d p(3,3);
std::cout << p << std::endl;

```

```

(3,3)

```

3.4.2 Site sets

Site set

3.4.2.1 Creating a site set

```

box2d b(4,4);

(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3),
(3,0), (3,1), (3,2), (3,3),

mln_piter_(p_array<point2d>) p(arr);
for_all(p)
    std::cout << p << ", ";
std::cout << std::endl;

(-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2),
(0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-2), (1,-1), (1,0), (1,1), (1,2), (2,-2),
(2,-1), (2,0), (2,1), (2,2),

mln_piter_(box2d) p(b);
for_all(p)
    std::cout << p << ", ";
std::cout << std::endl;

(2,2), (1,2),

```

3.4.2.2 Getting access to sites

3.4.3 Images

3.4.3.1 Creating an image

3.4.3.2 Reading an image from a file

3.4.3.3 Accessing data

[Getting started with Milena Load and save images](#)

3.5 Load and save images

```
image2d<bool> ima;
io::pbm::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbm::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");
```

[Load and save images Data representation Create your first image](#)

3.6 Create your first image

See also

[tuto2_first_image.cc](#)

```
bool vals[13][21] = {
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};
```

```
image2d<bool> ima = make::image(vals);
```

[Create an image](#)

```
debug::println(ima);
```

```
- - - - -
- | - | - | | - | - - - | - - - - | - -
- | - | - | - - - | - - - | - - - | - | -
- | | | - | | | - | - - - | - - - | - | -
- | - | - | - - - | - - - | - - - | - | -
```

```

- | - | - | | - | | | - | | | - - | - -
- - - - - - - - - - - - - - - - - - - -
- | - | - - | - - | | - - | - - - | | - -
- | - | - | - | - | - | - | - - - | - | -
- | - | - | - | - | | - - | - - - | - | -
- | | | - | - | - | - | - | - - - | - | -
- | - | - - | - - | - | - | | | - - -
- - - - - - - - - - - - - - - - - - - -

```

```
doc::pbmsave(ima, "tuto2_first_image");
```

[Possible value types](#) [Load and save images](#) [Read and write images](#)

3.7 Read and write images

See also

[tuto3_rw_image.cc](#)

```

image2d<value::rgb8> ima(40, 40);

data::fill(ima, literal::red);

for (def::coord row = 20; row < 30; ++row)
    for (def::coord col = 20; col < 30; ++col)
        ima(point2d(row, col)) = literal::blue;

for (def::coord row = 20; row < 30; ++row)
    for (def::coord col = 20; col < 30; ++col)
        opt::at(ima, row, col) = literal::blue;

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

data::fill(ima, lena);

data::paste(ima, lena);

```

[Access and modify values](#) [Fill](#) [Paste](#) [Create your first image](#) [Regions of interest](#)

3.8 Regions of interest

See also

[tuto4_genericity_and_algorithms.cc](#)

```

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

namespace data
{
    template <typename I, typename D>
    void fill(Image<I>& ima, const D& data);
}

```

3.8.1 Image domain restricted by a site set

```
box2d roi = make::box2d(20, 20, 39, 39);

data::fill((lena | roi).rw(), literal::green);
```

3.8.2 Image domain restricted by a function

```
p_array<point2d> arr;
for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if ((row + col) % 2 == 0)
      arr.append(point2d(row, col));

for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if ((row + col) % 2 == 0)
      opt::at(lena, row, col) = literal::green;

data::fill((lena | fun::p2b::chess()).rw(), literal::green);
```

3.8.3 Image domain restricted by a mask

```
image2d<bool> mask;
initialize(mask, lena);
data::fill(mask, false);
data::fill((mask | make::box2d(10, 10, 14, 14)).rw(), true);
data::fill((mask | make::box2d(25, 15, 29, 18)).rw(), true);
data::fill((mask | make::box2d(50, 50, 54, 54)).rw(), true);

data::fill((lena | pw::value(mask)).rw(), literal::green);
```

3.8.4 Image domain restricted by a predicate

```
image2d<bool> lena_bw = binarization::binarization(lena, keep_specific_colors()
);
value::label_8 nlabels;
image2d<value::label_8> label = labeling::blobs(lena_bw, c8(), nlabels);

data::fill((lena | (pw::value(label) == pw::cst(0u))).rw(), literal::blue);
```

[Read and write images](#)

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Types	98
Graphes	94
Images	95
Basic types	95
Image morphers	96
Values morphers	96
Domain morphers	97
Identity morphers	97
Neighborhoods	101
1D neighborhoods	101
2D neighborhoods	102
3D neighborhoods	104
Site sets	108
Basic types	108
Graph based	109
Complex based	109
Sparse types	109
Queue based	110
Utilities	110
Windows	111
1D windows	112
2D windows	113
3D windows	115
N-D windows	118
Multiple windows	118
Accumulators	99
On site sets	91
On images	91
On values	92
Multiple accumulators	94
Routines	99
Canvas	99

Functions	99
v2w2v functions	118
v2w_w2v functions	119
vv2b functions	119

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

mln (Mln/convert/to_image.hh)	121
mln::accu (Namespace of accumulators)	162
mln::accu::image (Namespace of accumulator image routines)	166
mln::accu::impl (Implementation namespace of accumulator namespace)	166
mln::accu::logic (Namespace of logical accumulators)	166
mln::accu::math (Namespace of mathematic accumulators)	167
mln::accu::meta::logic (Namespace of logical meta-accumulators)	167
mln::accu::meta::math (Namespace of mathematic meta-accumulators)	168
mln::accu::meta::shape (Namespace of shape meta-accumulators)	168
mln::accu::meta::stat (Namespace of statistical meta-accumulators)	169
mln::accu::shape (Namespace of shape accumulators)	170
mln::accu::stat (Namespace of statistical accumulators)	170
mln::algebra (Namespace of algebraic structure)	172
mln::arith (Namespace of arithmetic)	173
mln::arith::impl (Implementation namespace of arith namespace)	186
mln::arith::impl::generic (Generic implementation namespace of arith namespace)	186
mln::binarization (Namespace of "point-wise" expression tools)	187
mln::border (Namespace of routines related to image virtual (outer) border)	188
mln::border::impl (Implementation namespace of border namespace)	191
mln::border::impl::generic (Generic implementation namespace of border namespace)	192
mln::canvas (Namespace of canvas)	192
mln::canvas::browsing (Namespace of browsing canvas)	193
mln::canvas::impl (Implementation namespace of canvas namespace)	194
mln::canvas::labeling (Namespace of labeling canvas)	194
mln::canvas::labeling::impl (Implementation namespace of labeling canvas namespace)	195
mln::canvas::morpho (Namespace of morphological canvas)	195
mln::convert (Namespace of conversion routines)	196
mln::data (Namespace of image processing routines related to pixel data)	202
mln::data::approx (Namespace of image processing routines related to pixel levels with approxi- mation)	214
mln::data::approx::impl (Implementation namespace of data::approx namespace)	215
mln::data::impl (Implementation namespace of data namespace)	215
mln::data::impl::generic (Generic implementation namespace of data namespace)	218

mln::data::naive (Namespace of image processing routines related to pixel levels with naive approach)	221
mln::data::naive::impl (Implementation namespace of data::naive namespace)	222
mln::debug (Namespace of routines that help to debug)	223
mln::debug::impl (Implementation namespace of debug namespace)	228
mln::def (Namespace for core definitions)	229
mln::display (Namespace of routines that help to display images)	230
mln::display::impl (Implementation namespace of display namespace)	230
mln::display::impl::generic (Generic implementation namespace of display namespace)	230
mln::doc (The namespace mln::doc is only for documentation purpose)	230
mln::draw (Namespace of drawing routines)	232
mln::estim (Namespace of estimation materials)	235
mln::extension (Namespace of extension tools)	237
mln::fun (Namespace of functions)	240
mln::fun::access (Namespace for access functions)	241
mln::fun::i2v (Namespace of integer-to-value functions)	242
mln::fun::n2v (Namespace of functions from nil to value)	242
mln::fun::p2b (Namespace of functions from point to boolean)	242
mln::fun::p2p (Namespace of functions from grid point to grid point)	243
mln::fun::p2v (Namespace of functions from point to value)	243
mln::fun::stat (Namespace of statistical functions)	243
mln::fun::v2b (Namespace of functions from value to logic value)	243
mln::fun::v2i (Namespace of value-to-integer functions)	244
mln::fun::v2v (Namespace of functions from value to value)	244
mln::fun::v2w2v (Namespace of bijective functions)	246
mln::fun::v2w_w2v (Namespace of functions from value to value)	246
mln::fun::vv2b (Namespace of functions from value to value)	246
mln::fun::vv2v (Namespace of functions from a couple of values to a value)	247
mln::fun::x2p (Namespace of functions from point to value)	248
mln::fun::x2v (Namespace of functions from vector to value)	248
mln::fun::x2x (Namespace of functions from vector to vector)	248
mln::geom (Namespace of all things related to geometry)	249
mln::geom::impl (Implementation namespace of geom namespace)	262
mln::graph (Namespace of graph related routines)	263
mln::grid (Namespace of grids definitions)	265
mln::histo (Namespace of histograms)	266
mln::histo::impl (Implementation namespace of histo namespace)	267
mln::histo::impl::generic (Generic implementation namespace of histo namespace)	267
mln::impl (Implementation namespace of mln namespace)	267
mln::io (Namespace of input/output handling)	267
mln::io::cloud (Namespace of cloud input/output handling)	269
mln::io::dicom (Namespace of DICOM input/output handling)	270
mln::io::dump (Namespace of dump input/output handling)	271
mln::io::fits (Namespace of fits input/output handling)	272
mln::io::fld (Namespace of pgm input/output handling)	273
mln::io::magick (Namespace of magick input/output handling)	274
mln::io::off (Namespace of off input/output handling)	276
mln::io::pbm (Namespace of pbm input/output handling)	277
mln::io::pbm::impl (Namespace of pbm implementation details)	279
mln::io::pbms (Namespace of pbms input/output handling)	279
mln::io::pbms::impl (Namespace of pbms implementation details)	280
mln::io::pfm (Namespace of pfm input/output handling)	280
mln::io::pfm::impl (Implementation namespace of pfm namespace)	281
mln::io::pgm (Namespace of pgm input/output handling)	282

mln::io::pgms (Namespace of pgms input/output handling)	283
mln::io::plot (Namespace of plot input/output handling)	283
mln::io::pnm (Namespace of pnm input/output handling)	285
mln::io::pnm::impl (Namespace of pnm's implementation details)	287
mln::io::pnms (Namespace of pnms input/output handling)	287
mln::io::ppm (Namespace of ppm input/output handling)	288
mln::io::ppms (Namespace of ppms input/output handling)	289
mln::io::raw (Namespace of raw input/output handling)	290
mln::io::tiff (Namespace of tiff input/output handling)	291
mln::io::txt (Namespace of txt input/output handling)	292
mln::labeling (Namespace of labeling routines)	292
mln::labeling::impl (Implementation namespace of labeling namespace)	308
mln::labeling::impl::generic (Generic implementation namespace of labeling namespace)	309
mln::linear (Namespace of linear image processing routines)	311
mln::linear::impl (Namespace of linear image processing routines implementation details)	315
mln::linear::local (Specializations of local linear routines)	315
mln::linear::local::impl (Namespace of local linear routines implementation details)	317
mln::literal (Namespace of literals)	317
mln::logical (Namespace of logic)	323
mln::logical::impl (Implementation namespace of logical namespace)	326
mln::logical::impl::generic (Generic implementation namespace of logical namespace)	327
mln::make (Namespace of routines that help to make Milena's objects)	327
mln::math (Namespace of mathematical routines)	350
mln::metal (Namespace of meta-programming tools)	351
mln::metal::impl (Implementation namespace of metal namespace)	352
mln::metal::math (Namespace of static mathematical functions)	352
mln::metal::math::impl (Implementation namespace of metal::math namespace)	352
mln::morpho (Namespace of mathematical morphology routines)	353
mln::morpho::approx (Namespace of approximate mathematical morphology routines)	363
mln::morpho::attribute (Namespace of attributes used in mathematical morphology)	363
mln::morpho::closing::approx (Namespace of approximate mathematical morphology closing routines)	363
mln::morpho::elementary (Namespace of image processing routines of elementary mathematical morphology)	364
mln::morpho::impl (Namespace of mathematical morphology routines implementations)	366
mln::morpho::impl::generic (Namespace of mathematical morphology routines generic implementations)	366
mln::morpho::opening::approx (Namespace of approximate mathematical morphology opening routines)	367
mln::morpho::reconstruction (Namespace of morphological reconstruction routines)	367
mln::morpho::reconstruction::by_dilation (Namespace of morphological reconstruction by dilation routines)	368
mln::morpho::reconstruction::by_erosion (Namespace of morphological reconstruction by erosion routines)	368
mln::morpho::tree (Namespace of morphological tree-related routines)	368
mln::morpho::tree::filter (Namespace for attribute filtering)	375
mln::morpho::watershed (Namespace of morphological watershed routines)	378
mln::morpho::watershed::watershed (Namespace of morphological watershed routines implementations)	380
mln::morpho::watershed::watershed::generic (Namespace of morphological watershed routines generic implementations)	380
mln::norm (Namespace of norms)	381
mln::norm::impl (Implementation namespace of norm namespace)	383
mln::opt (Namespace of optional routines)	383

mln::opt::impl (Implementation namespace of opt namespace)	385
mln::pw (Namespace of "point-wise" expression tools)	385
mln::registration (Namespace of "point-wise" expression tools)	385
mln::select (Select namespace (FIXME doc))	388
mln::set (Namespace of image processing routines related to pixel sets)	388
mln::subsampling (Namespace of "point-wise" expression tools)	391
mln::tag (Namespace of image processing routines related to tags)	393
mln::test (Namespace of image processing routines related to pixel tests)	393
mln::test::impl (Implementation namespace of test namespace)	395
mln::topo (Namespace of "point-wise" expression tools)	395
mln::trace (Namespace of routines related to the trace mechanism)	405
mln::trait (Namespace where traits are defined)	405
mln::transform (Namespace of transforms)	405
mln::util (Namespace of tools using for more complex algorithm)	410
mln::util::impl (Implementation namespace of util namespace)	417
mln::value (Namespace of materials related to pixel value types)	417
mln::value::impl (Implementation namespace of value namespace)	429
mln::win (Namespace of image processing routines related to win)	429

Chapter 6

Class Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

mln::Generalized_Pixel< bkd_pixter1d< I > >	720
Pixel_Iterator< bkd_pixter1d< I > >	
pixel_iterator_base_< I, bkd_pixter1d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::bkd_pixter1d< I >	544
mln::Generalized_Pixel< bkd_pixter2d< I > >	720
Pixel_Iterator< bkd_pixter2d< I > >	
pixel_iterator_base_< I, bkd_pixter2d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::bkd_pixter2d< I >	545
mln::Generalized_Pixel< bkd_pixter3d< I > >	720
Pixel_Iterator< bkd_pixter3d< I > >	
pixel_iterator_base_< I, bkd_pixter3d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::bkd_pixter3d< I >	547
mln::Generalized_Pixel< dpoints_bkd_pixter< I > >	720
Pixel_Iterator< dpoints_bkd_pixter< I > >	
mln::dpoints_bkd_pixter< I >	644
mln::Generalized_Pixel< dpoints_fwd_pixter< I > >	720
Pixel_Iterator< dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I >	646
mln::Generalized_Pixel< fwd_pixter1d< I > >	720
Pixel_Iterator< fwd_pixter1d< I > >	
pixel_iterator_base_< I, fwd_pixter1d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::fwd_pixter1d< I >	714
mln::Generalized_Pixel< fwd_pixter2d< I > >	720
Pixel_Iterator< fwd_pixter2d< I > >	
pixel_iterator_base_< I, fwd_pixter2d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::fwd_pixter2d< I >	715

mln::Generalized_Pixel< fwd_pixter3d< I > >	720
Pixel_Iterator< fwd_pixter3d< I > >	
pixel_iterator_base_< I, fwd_pixter3d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::fwd_pixter3d< I >	717
mln::Generalized_Pixel< pixel< I > >	720
mln::pixel< I >	928
mln::internal::image_base< algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >	
image_morpher< I, algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >	
image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >	
mln::value::stack_image< n, I >	1123
mln::internal::image_base< const I::value, I::domain_t, E >	
image_morpher< const I, const I::value, I::domain_t, E >	
image_identity< const I, I::domain_t, E >	
mln::labeled_image_base< I, E >	787
mln::internal::image_base< const I::value, I::domain_t, labeled_image< I > >	
image_morpher< const I, const I::value, I::domain_t, labeled_image< I > >	
image_identity< const I, I::domain_t, labeled_image< I > >	
mln::labeled_image_base< I, labeled_image< I > >	787
mln::labeled_image< I >	783
mln::internal::image_base< F::result, I1::domain_t, thrubin_image< I1, I2, F > >	
image_morpher< I1, F::result, I1::domain_t, thrubin_image< I1, I2, F > >	
image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >	
mln::thrubin_image< I1, I2, F >	957
mln::internal::image_base< F::result, I::domain_t, fun_image< F, I > >	
image_morpher< I, F::result, I::domain_t, fun_image< F, I > >	
image_value_morpher< I, F::result, fun_image< F, I > >	
mln::fun_image< F, I >	707
mln::internal::image_base< F::result, S, E >	
image_primary< F::result, S, E >	
mln::internal::image_base< F::result, S, image< F, S > >	
image_primary< F::result, S, image< F, S > >	
image_base< F, S, image< F, S > >	
mln::pw::image< F, S >	942
mln::internal::image_base< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
mln::edge_image< P, V, G >	652
mln::internal::image_base< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
mln::vertex_image< P, V, G >	1128
mln::internal::image_base< I::value, box2d_h, hexa< I > >	
image_morpher< I, I::value, box2d_h, hexa< I > >	
image_domain_morpher< I, box2d_h, hexa< I > >	
mln::hexa< I >	753
mln::internal::image_base< I::value, box< I::site >, extended< I > >	

image_morpher< I, I::value, box< I::site >, extended< I > >	
image_domain_morpher< I, box< I::site >, extended< I > >	
mln::extended< I >	654
mln::internal::image_base< I::value, D, unproject_image< I, D, F > >	
image_morpher< I, I::value, D, unproject_image< I, D, F > >	
image_domain_morpher< I, D, unproject_image< I, D, F > >	
mln::unproject_image< I, D, F >	1017
mln::internal::image_base< I::value, I::domain_t, decorated_image< I, D > >	
image_morpher< I, I::value, I::domain_t, decorated_image< I, D > >	
image_identity< I, I::domain_t, decorated_image< I, D > >	
mln::decorated_image< I, D >	591
mln::internal::image_base< I::value, I::domain_t, extension_fun< I, F > >	
image_morpher< I, I::value, I::domain_t, extension_fun< I, F > >	
image_identity< I, I::domain_t, extension_fun< I, F > >	
mln::extension_fun< I, F >	656
mln::internal::image_base< I::value, I::domain_t, extension_ima< I, J > >	
image_morpher< I, I::value, I::domain_t, extension_ima< I, J > >	
image_identity< I, I::domain_t, extension_ima< I, J > >	
mln::extension_ima< I, J >	658
mln::internal::image_base< I::value, I::domain_t, extension_val< I > >	
image_morpher< I, I::value, I::domain_t, extension_val< I > >	
image_identity< I, I::domain_t, extension_val< I > >	
mln::extension_val< I >	661
mln::internal::image_base< I::value, I::domain_t, interpolated< I, F > >	
image_morpher< I, I::value, I::domain_t, interpolated< I, F > >	
image_identity< I, I::domain_t, interpolated< I, F > >	
mln::interpolated< I, F >	777
mln::internal::image_base< I::value, I::domain_t, p2p_image< I, F > >	
image_morpher< I, I::value, I::domain_t, p2p_image< I, F > >	
image_domain_morpher< I, I::domain_t, p2p_image< I, F > >	
mln::p2p_image< I, F >	837
mln::internal::image_base< I::value, I::domain_t, plain< I > >	
image_morpher< I, I::value, I::domain_t, plain< I > >	
image_identity< I, I::domain_t, plain< I > >	
mln::plain< I >	930
mln::internal::image_base< I::value, I::domain_t, safe_image< I > >	
image_morpher< I, I::value, I::domain_t, safe_image< I > >	
image_identity< I, I::domain_t, safe_image< I > >	
mln::safe_image< I >	945
mln::internal::image_base< I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_morpher< I, I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
mln::sub_image_if< I, S >	955
mln::internal::image_base< I::value, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
image_morpher< I, I::value, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
image_domain_morpher< I, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
mln::transformed_image< I, F >	1015
mln::internal::image_base< I::value, S, E >	
image_morpher< I, I::value, S, E >	
mln::internal::image_base< I::value, S, sub_image< I, S > >	
image_morpher< I, I::value, S, sub_image< I, S > >	
image_domain_morpher< I, S, sub_image< I, S > >	
mln::sub_image< I, S >	953


```

mln::internal::image_base< I::value, S, tr_image< S, I, T > >
  image_morpher< I, I::value, S, tr_image< S, I, T > >
  image_identity< I, S, tr_image< S, I, T > >
  mln::tr_image< S, I, T > . . . . . 1012
mln::internal::image_base< image2d< V >::value, box2d_h, hexa< image2d< V > > >
  image_morpher< image2d< V >, image2d< V >::value, box2d_h, hexa< image2d< V > > >
  image_domain_morpher< image2d< V >, box2d_h, hexa< image2d< V > > >
  mln::hexa< image2d< V > > . . . . . 753
  mln::image2d_h< V > . . . . . 769
mln::internal::image_base< mln::trait::ch_value< I, F::result >::ret::value, I::domain_t, lazy_image<
I, F, B > >
  image_morpher< mln::trait::ch_value< I, F::result >::ret, mln::trait::ch_value< I, F::result
>::ret::value, I::domain_t, lazy_image< I, F, B > >
  image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B >
>
  mln::lazy_image< I, F, B > . . . . . 790
mln::internal::image_base< T, box1d, image1d< T > >
  image_primary< T, box1d, image1d< T > >
  mln::image1d< T > . . . . . 759
mln::internal::image_base< T, box3d, image3d< T > >
  image_primary< T, box3d, image3d< T > >
  mln::image3d< T > . . . . . 772
mln::internal::image_base< T, I::domain_t, E >
  image_morpher< I, T, I::domain_t, E >
mln::internal::image_base< T, I::domain_t, violent_cast_image< T, I > >
  image_morpher< I, T, I::domain_t, violent_cast_image< T, I > >
  image_value_morpher< I, T, violent_cast_image< T, I > >
  mln::violent_cast_image< T, I > . . . . . 1130
mln::internal::image_base< T, mln::box2d, image2d< T > >
  image_primary< T, mln::box2d, image2d< T > >
  mln::image2d< T > . . . . . 764
mln::internal::image_base< T, S, flat_image< T, S > >
  image_primary< T, S, flat_image< T, S > >
  mln::flat_image< T, S > . . . . . 663
mln::internal::image_base< V, p_complex< D, G >, complex_image< D, G, V > >
  image_primary< V, p_complex< D, G >, complex_image< D, G, V > >
  mln::complex_image< D, G, V > . . . . . 579
mln::internal::check::image_fastest< complex_image< D, G, V >, mln::metal::equal< mln_trait_
image_speed(complex_image< D, G, V >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< decorated_image< I, D >, mln::metal::equal< mln_trait_
image_speed(decorated_image< I, D >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< E, mln::metal::equal< mln_trait_image_speed(E),
trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< edge_image< P, V, G >, mln::metal::equal< mln_trait_image_
speed(edge_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extended< I >, mln::metal::equal< mln_trait_image_
speed(extended< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extension_fun< I, F >, mln::metal::equal< mln_trait_image_
speed(extension_fun< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extension_ima< I, J >, mln::metal::equal< mln_trait_image_
speed(extension_ima< I, J >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extension_val< I >, mln::metal::equal< mln_trait_image_
speed(extension_val< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< flat_image< T, S >, mln::metal::equal< mln_trait_image_

```



```

speed(flat_image< T, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< fun_image< F, I >, mln::metal::equal< mln_trait_image_
speed(fun_image< F, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< I >, mln::metal::equal< mln_trait_image_speed(hexa<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< image2d< V > >, mln::metal::equal< mln_trait_
image_speed(hexa< image2d< V > >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image1d< T >, mln::metal::equal< mln_trait_image_
speed(image1d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image2d< T >, mln::metal::equal< mln_trait_image_
speed(image2d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< T >, mln::metal::equal< mln_trait_image_
speed(image3d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image< F, S >, mln::metal::equal< mln_trait_image_
speed(image< F, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< interpolated< I, F >, mln::metal::equal< mln_trait_image_
speed(interpolated< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< labeled_image< I >, mln::metal::equal< mln_trait_image_
speed(labeled_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< lazy_image< I, F, B >, mln::metal::equal< mln_trait_image_
speed(lazy_image< I, F, B >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< p2p_image< I, F >, mln::metal::equal< mln_trait_image_
speed(p2p_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< plain< I >, mln::metal::equal< mln_trait_image_speed(plain<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< safe_image< I >, mln::metal::equal< mln_trait_image_
speed(safe_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< stack_image< n, I >, mln::metal::equal< mln_trait_image_
speed(stack_image< n, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image_if< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image_if< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< thrubin_image< I1, I2, F >, mln::metal::equal< mln_trait_
image_speed(thrubin_image< I1, I2, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< tr_image< S, I, T >, mln::metal::equal< mln_trait_image_
speed(tr_image< S, I, T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< transformed_image< I, F >, mln::metal::equal< mln_trait_
image_speed(transformed_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< unproject_image< I, D, F >, mln::metal::equal< mln_trait_
image_speed(unproject_image< I, D, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< vertex_image< P, V, G >, mln::metal::equal< mln_trait_
image_speed(vertex_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< violent_cast_image< T, I >, mln::metal::equal< mln_trait_
image_speed(violent_cast_image< T, I >), trait::image::speed::fastest >::eval >
mln::internal::impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >
    mln::graph_window_piter< S, W, I > . . . . . 750
mln::value::Integer< graylevel< n > > . . . . . 1107
    mln::value::graylevel< n > . . . . . 1094
mln::value::Integer< int_s< n > > . . . . . 1107
    mln::value::int_s< n > . . . . . 1100
mln::value::Integer< int_u< n > > . . . . . 1107
    mln::value::int_u< n > . . . . . 1103

```

mln::value::Integer< int_u_sat< n > >	1107
mln::value::int_u_sat< n >	1105
mln::value::Integer< object_id< Tag, V > >	1107
mln::util::object_id< Tag, V >	1057
mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >	
mln::graph_window_if_piter< S, W, I >	748
mln::algebra::h_mat< d, T >	540
mln::algebra::h_vec< d, C >	542
mln::canvas::chamfer< F >	578
mln::category< R(*) (A) >	578
mln::Delta_Point_Site< void >	595
mln::doc::Accumulator< E >	595
mln::doc::Generalized_Pixel< E >	609
mln::doc::Pixel_Iterator< E >	620
mln::doc::Object< E >	620
mln::doc::Dpoint< E >	599
mln::doc::Image< E >	611
mln::doc::Fastest_Image< E >	601
mln::doc::Iterator< E >	617
mln::doc::Pixel_Iterator< E >	620
mln::doc::Site_Iterator< E >	626
mln::doc::Value_Iterator< E >	630
mln::doc::Neighborhood< E >	618
mln::doc::Site_Set< E >	628
mln::doc::Box< E >	596
mln::doc::Value_Set< E >	631
mln::doc::Weighted_Window< E >	634
mln::doc::Window< E >	636
mln::doc::Point_Site< E >	623
mln::Edge< E >	652
mln::fun::from_accu< A >	666
mln::fun::internal::ch_function_value_impl< F, V >	
mln::fun::v2v::ch_function_value< F, V >	672
mln::fun::x2p::closest_point< P >	697
mln::fun::x2x::composed< T2, T1 >	699
mln::Function< void >	710
mln::Gdpoint< void >	719
mln::Generalized_Pixel< E >	720
mln::Pixel_Iterator	
mln::internal::pixel_iterator_base_	
mln::geom::complex_geometry< D, P >	720
mln::graph::attribute::card_t	727
mln::graph::attribute::representative_t	727
mln::histo::array< T >	757
mln::internal::image_base< T, S, E >	
mln::internal::neighborhood_base< W, E >	
mln::internal::neighb_base	
mln::neighb< graph_elt_mixed_window< G, S, S2 > >	835
mln::graph_elt_mixed_neighborhood< G, S, S2 >	728
mln::neighb< graph_elt_window< G, S > >	835
mln::graph_elt_neighborhood< G, S >	733

mln::neighb< graph_elt_window_if< G, S, I > >	835
mln::graph_elt_neighborhood_if< G, S, I >	735
mln::internal::pixel_impl< I, E >	
mln::internal::pixel_iterator_base_	
mln::io::dicom::dicom_header	780
mln::io::dump::dump_header	780
mln::io::fld::fld_header	780
mln::io::raw::raw_header	780
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >	822
mln::metal::bool_< false >	
mln::metal::equal< T1::coord, T2::coord >	822
mln::metal::equal< T1::point, T2::point >	822
mln::metal::equal< T1, T2 >	822
mln::metal::converts_to< T, U >	822
mln::metal::goes_to< T, U >	823
mln::metal::is< T, U >	823
mln::metal::is_a< T, M >	824
mln::metal::is_not< T, U >	824
mln::metal::is_not_a< T, M >	824
mln::Neighborhood< void >	837
mln::Object< E >	837
mln::Browsing< E >	564
mln::Delta_Point_Site< E >	594
mln::Dpoint< E >	638
mln::Function< E >	710
mln::Function_n2v< E >	711
mln::Function_v2v< E >	712
mln::fun::x2v::bilinear< I >	698
mln::fun::x2v::trilinear< I >	699
mln::fun::x2x::linear< I >	700
mln::Function_v2b< E >	711
mln::Function_vv2b< E >	713
mln::Function_vv2v< E >	713
mln::Gdpoint< E >	718
mln::Graph< E >	726
mln::Image< E >	757
mln::Iterator< E >	781
mln::Pixel_Iterator	
mln::topo::internal::complex_iterator_base	
mln::topo::internal::complex_relative_iterator_base	
mln::Literal< E >	793
mln::Mesh< E >	817
mln::Regular_Grid< E >	944
mln::Meta_Accumulator< E >	818
mln::Meta_Function< E >	820
mln::Meta_Function_v2v< E >	821
mln::Meta_Function_vv2v< E >	821
mln::Neighborhood< E >	836
mln::Proxy< E >	941
mln::Accumulator< E >	538
mln::accu::internal::base	
couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P >	
>	

mln::accu::site_set::rectangularity< P >	505
mln::accu::pair< min< V >, max< V > >	495
mln::accu::stat::min_max< V >	523
mln::Site_Proxy< E >	949
mln::Pseudo_Site< E >	941
mln::Site_Iterator< E >	947
mln::internal::site_iterator_base	
mln::internal::site_set_iterator_base	
mln::p_transformed_piter< Pi, S, F >	916
mln::Site< E >	946
mln::Gpoint< E >	722
mln::Site_Set< E >	950
mln::Box< E >	556
mln::Value< E >	1089
mln::Weighted_Window< E >	1135
mln::Window< E >	1148
mln::graph_window_base< P, E >	746
mln::internal::window_base	
mln::Proxy< void >	941
mln::Pseudo_Site< void >	942
mln::registration::closest_point_basic< P >	943
mln::registration::closest_point_with_map< P >	944
mln::select::p_of< P >	945
mln::Site< void >	947
mln::Site_Proxy< void >	950
mln::Site_Set< void >	953
mln::thru_image< I, F >	956
mln::topo::complex< D >	986
mln::topo::face< D >	989
mln::topo::algebraic_face< D >	973
mln::topo::is_simple_2d_t< N >	997
mln::topo::n_face< N, D >	1001
mln::topo::algebraic_n_face< N, D >	978
mln::topo::n_faces_set< N, D >	1007
mln::topo::skeleton::is_simple_point< N >	1009
mln::util::adjacency_matrix< V >	1019
mln::util::array< T >	1019
mln::util::branch< T >	1027
mln::util::branch_iter< T >	1028
mln::util::branch_iter_ind< T >	1030
mln::util::greater_point< I >	1048
mln::util::greater_psite< I >	1049
mln::util::head< T, R >	1049
mln::util::ilcell< T >	1050
mln::util::internal::edge_impl< G >	
mln::util::edge< G >	1034
mln::util::internal::vertex_impl< G >	
mln::util::vertex< G >	1084
mln::util::node< T, R >	1057
mln::util::ord< T >	1059
mln::util::pix< I >	1062
mln::util::tracked_ptr< T >	1075

mln::util::tree< T >	1077
mln::util::tree_node< T >	1079
mln::value::float01	1091
mln::value::Integer< E >	1107
mln::value::Integer< void >	1108
mln::value::internal::value_like_< V, C, N, E >	
mln::value::float01_f	1093
mln::value::graylevel< n >	1094
mln::value::graylevel_f	1098
mln::value::int_s< n >	1100
mln::value::int_u< n >	1103
mln::value::int_u_sat< n >	1105
mln::value::label< n >	1108
mln::value::qt::rgb32	1116
mln::value::rgb< n >	1118
mln::value::set< T >	1120
mln::value::sign	1121
mln::value::super_value< sign >	1126
mln::value::value_array< T, V >	1126
mln::Vertex< E >	1127
mln::internal::neighborhood_base< W, neighb< W > >	
neighb_base< W, neighb< W > >	
mln::neighb< W >	835
mln::Object< abs >	837
mln::Meta_Function< abs >	820
mln::Meta_Function_v2v< abs >	821
mln::Object< abs< V > >	837
mln::Function< abs< V > >	710
mln::Function_v2v< abs< V > >	712
mln::Object< accu_result >	837
mln::Meta_Function< accu_result >	820
mln::Meta_Function_v2v< accu_result >	821
mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > >	837
mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	
mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >	959
mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D > >	837
mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >	960
mln::Object< adj_higher_face_bkd_iter< D > >	837
mln::Iterator< adj_higher_face_bkd_iter< D > >	781

complex_iterator_base< algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
mln::topo::adj_higher_face_bkd_iter< D >	961
mln::Object< adj_higher_face_fwd_iter< D > >	837
mln::Iterator< adj_higher_face_fwd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
mln::topo::adj_higher_face_fwd_iter< D >	962
mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D > >	837
mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >	963
mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D > >	837
mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >	964
mln::Object< adj_lower_face_bkd_iter< D > >	837
mln::Iterator< adj_lower_face_bkd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
mln::topo::adj_lower_face_bkd_iter< D >	965
mln::Object< adj_lower_face_fwd_iter< D > >	837
mln::Iterator< adj_lower_face_fwd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_lower_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter< D > >	
mln::topo::adj_lower_face_fwd_iter< D >	967
mln::Object< adj_lower_higher_face_bkd_iter< D > >	837
mln::Iterator< adj_lower_higher_face_bkd_iter< D > >	781
complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D > >	

mln::topo::adj_lower_higher_face_bkd_iter< D >	968
mln::Object< adj_lower_higher_face_fwd_iter< D > >	837
mln::Iterator< adj_lower_higher_face_fwd_iter< D > >	781
complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D >, adj_lower_higher_face_fwd_iter< D > >	
mln::topo::adj_lower_higher_face_fwd_iter< D >	969
mln::Object< adj_m_face_bkd_iter< D > >	837
mln::Iterator< adj_m_face_bkd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_m_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D > >	
mln::topo::adj_m_face_bkd_iter< D >	970
mln::Object< adj_m_face_fwd_iter< D > >	837
mln::Iterator< adj_m_face_fwd_iter< D > >	781
complex_iterator_base< algebraic_face< D >, adj_m_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >	
mln::topo::adj_m_face_fwd_iter< D >	971
mln::Object< all_to< T > >	837
mln::Function< all_to< T > >	710
mln::Function_v2v< all_to< T > >	712
mln::Object< antilogy >	837
mln::Function< antilogy >	710
mln::Function_v2v< antilogy >	712
mln::Function_v2b< antilogy >	711
mln::fun::p2b::antilogy	667
mln::Object< array1d< T, Size > >	837
mln::Object< array2d< T, r, c > >	837
mln::Object< array3d< T, s, r, c > >	837
mln::Object< array_bkd_iter< T > >	837
mln::Proxy< array_bkd_iter< T > >	941
mln::Object< array_fwd_iter< T > >	837
mln::Proxy< array_fwd_iter< T > >	941
mln::Object< asc_propagation >	837
mln::Object< backdiag2d >	837
mln::Window< backdiag2d >	1148
window_base< dpoint2d, backdiag2d >	
classical_window_base< dpoint2d, backdiag2d >	
mln::win::backdiag2d	1137
mln::Object< backdiagonal2d_t >	837
mln::Browsing< backdiagonal2d_t >	564
mln::canvas::browsing::backdiagonal2d_t	565
mln::Object< ball< G, C > >	837
mln::Window< ball< G, C > >	1148
window_base< dpoint< G, C >, ball< G, C > >	
classical_window_base< dpoint< G, C >, ball< G, C > >	

mln::win::ball< G, C >	1138
mln::Object< bbox >	837
mln::Meta_Accumulator< bbox >	818
mln::accu::meta::shape::bbox	478
mln::Object< bbox< P > >	837
mln::Proxy< bbox< P > >	941
mln::Accumulator< bbox< P > >	538
base< const box< P > &, bbox< P > >	
mln::accu::shape::bbox< P >	500
mln::Object< big_chess< B > >	837
mln::Function< big_chess< B > >	710
mln::Function_v2v< big_chess< B > >	712
mln::Function_v2b< big_chess< B > >	711
mln::Object< bin_off_loader >	837
mln::Object< bin_off_saver >	837
mln::Object< binary< Fun, T1, T2 > >	837
mln::Function< binary< Fun, T1, T2 > >	710
mln::Function_v2v< binary< Fun, T1, T2 > >	712
mln::Object< bkd_pixter1d< I > >	837
mln::Iterator< bkd_pixter1d< I > >	781
Pixel_Iterator< bkd_pixter1d< I > >	
mln::Object< bkd_pixter2d< I > >	837
mln::Iterator< bkd_pixter2d< I > >	781
Pixel_Iterator< bkd_pixter2d< I > >	
mln::Object< bkd_pixter3d< I > >	837
mln::Iterator< bkd_pixter3d< I > >	781
Pixel_Iterator< bkd_pixter3d< I > >	
mln::Object< black_t >	837
mln::Literal< black_t >	793
mln::literal::black_t	795
mln::Object< blue >	837
mln::Meta_Function< blue >	820
mln::Meta_Function_v2v< blue >	821
mln::Object< blue_t >	837
mln::Literal< blue_t >	793
mln::literal::blue_t	796
mln::Object< box< P > >	837
mln::Site_Set< box< P > >	950
mln::Box< box< P > >	556
mln::box< P >	548
mln::Object< box_runend_piter< P > >	837
mln::Proxy< box_runend_piter< P > >	941
mln::Site_Proxy< box_runend_piter< P > >	949
mln::Site_Iterator< box_runend_piter< P > >	947
site_iterator_base< box< P >, box_runend_piter< P > >	
site_set_iterator_base< box< P >, box_runend_piter< P > >	
mln::box_runend_piter< P >	562
mln::Object< box_runstart_piter< P > >	837

mln::Proxy< box_runstart_piter< P > >	941
mln::Site_Proxy< box_runstart_piter< P > >	949
mln::Site_Iterator< box_runstart_piter< P > >	947
site_iterator_base< box< P >, box_runstart_piter< P > >	
site_set_iterator_base< box< P >, box_runstart_piter< P > >	
mln::box_runstart_piter< P >	563
mln::Object< breadth_first_search_t >	837
mln::Browsing< breadth_first_search_t >	564
graph_first_search_t< breadth_first_search_t, std::queue >	
mln::canvas::browsing::breadth_first_search_t	566
mln::Object< brown_t >	837
mln::Literal< brown_t >	793
mln::literal::brown_t	796
mln::Object< card< I > >	837
mln::Proxy< card< I > >	941
mln::Accumulator< card< I > >	538
base< unsigned, card< I > >	
mln::morpho::attribute::card< I >	824
mln::Object< cast< V > >	837
mln::Function< cast< V > >	710
mln::Function_v2v< cast< V > >	712
mln::Object< center >	837
mln::Meta_Accumulator< center >	818
mln::accu::meta::center	459
mln::Object< center< P, V > >	837
mln::Proxy< center< P, V > >	941
mln::Accumulator< center< P, V > >	538
base< V, center< P, V > >	
mln::accu::center< P, V >	433
mln::Object< center_only_iter< D > >	837
mln::Iterator< center_only_iter< D > >	781
complex_iterator_base< algebraic_face< D >, center_only_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_iter< D > >	
mln::topo::center_only_iter< D >	982
mln::Object< centered_bkd_iter_adapter< D, I > >	837
mln::Iterator< centered_bkd_iter_adapter< D, I > >	781
complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I > >	
mln::topo::centered_bkd_iter_adapter< D, I >	984
mln::Object< centered_fwd_iter_adapter< D, I > >	837
mln::Iterator< centered_fwd_iter_adapter< D, I > >	781
complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I > >	
mln::topo::centered_fwd_iter_adapter< D, I >	985
mln::Object< ch_function_value< F, V > >	837
mln::Function< ch_function_value< F, V > >	710
mln::Function_v2v< ch_function_value< F, V > >	712

mln::fun::v2v::ch_function_value< F, V >	672
mln::Object< ch_piter_image< I, Fwd > >	837
mln::Image< ch_piter_image< I, Fwd > >	757
mln::Object< chess >	837
mln::Function< chess >	710
mln::Function_v2v< chess >	712
mln::Function_v2b< chess >	711
mln::Object< col >	837
mln::Meta_Function< col >	820
mln::Meta_Function_v2v< col >	821
mln::Object< colorize >	837
mln::Function< colorize >	710
mln::Function_v2v< colorize >	712
mln::Object< comp >	837
mln::Meta_Function< comp >	820
mln::Meta_Function_v2v< comp >	821
mln::Object< comp_count >	837
mln::Meta_Function< comp_count >	820
mln::Meta_Function_v2v< comp_count >	821
mln::Object< complex_image< D, G, V > >	837
mln::Image< complex_image< D, G, V > >	757
mln::Object< complex_neighborhood_bkd_piter< I, G, N > >	837
mln::Proxy< complex_neighborhood_bkd_piter< I, G, N > >	941
mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N > >	949
mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > >	947
site_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >	
site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >	
mln::complex_neighborhood_bkd_piter< I, G, N >	582
mln::Object< complex_neighborhood_fwd_piter< I, G, N > >	837
mln::Proxy< complex_neighborhood_fwd_piter< I, G, N > >	941
mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N > >	949
mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >	947
site_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >	
site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >	
mln::complex_neighborhood_fwd_piter< I, G, N >	583
mln::Object< complex_psite< D, G > >	837
mln::Proxy< complex_psite< D, G > >	941
mln::Site_Proxy< complex_psite< D, G > >	949
mln::Pseudo_Site< complex_psite< D, G > >	941
pseudo_site_base< const G::site &, complex_psite< D, G > >	
mln::complex_psite< D, G >	585
mln::Object< complex_window_bkd_piter< I, G, W > >	837
mln::Proxy< complex_window_bkd_piter< I, G, W > >	941
mln::Site_Proxy< complex_window_bkd_piter< I, G, W > >	949
mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >	947
site_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
mln::complex_window_bkd_piter< I, G, W >	588
mln::Object< complex_window_fwd_piter< I, G, W > >	837

mln::Proxy< complex_window_fwd_piter< I, G, W > >	941
mln::Site_Proxy< complex_window_fwd_piter< I, G, W > >	949
mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >	947
site_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
mln::complex_window_fwd_piter< I, G, W >	590
mln::Object< component< T, i > >	837
mln::Function< component< T, i > >	710
mln::Function_v2v< component< T, i > >	712
mln::fun::v2v::component< T, i >	672
mln::Object< compose >	837
mln::Meta_Function< compose >	820
mln::Meta_Function_vv2v< compose >	821
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . .	837
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > >	820
mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > >	821
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . .	837
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > >	820
mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > >	821
mln::Object< concrete >	837
mln::Object< convert< V > >	837
mln::Function< convert< V > >	710
mln::Function_v2v< convert< V > >	712
mln::Object< convolve< T1, T2, R > >	837
mln::Proxy< convolve< T1, T2, R > >	941
mln::Accumulator< convolve< T1, T2, R > >	538
base< R, convolve< T1, T2, R > >	
mln::accu::convolve< T1, T2, R >	435
mln::Object< cos >	837
mln::Meta_Function< cos >	820
mln::Meta_Function_v2v< cos >	821
mln::Object< cos< V > >	837
mln::Function< cos< V > >	710
mln::Function_v2v< cos< V > >	712
mln::fun::v2w2v::cos< V >	679
mln::Object< count >	837
mln::Meta_Accumulator< count >	818
mln::accu::meta::math::count	469
mln::Object< count< T > >	837
mln::Proxy< count< T > >	941
mln::Accumulator< count< T > >	538
base< unsigned, count< T > >	
mln::accu::math::count< T >	451
mln::Object< count_adjacent_vertices >	837
mln::Meta_Accumulator< count_adjacent_vertices >	818

mln::accu::meta::count_adjacent_vertices	459
mln::Object< count_adjacent_vertices< F, S > >	837
mln::Proxy< count_adjacent_vertices< F, S > >	941
mln::Accumulator< count_adjacent_vertices< F, S > >	538
base< unsigned, count_adjacent_vertices< F, S > >	
mln::accu::count_adjacent_vertices< F, S >	436
mln::Object< count_adjacent_vertices< I > >	837
mln::Proxy< count_adjacent_vertices< I > >	941
mln::Accumulator< count_adjacent_vertices< I > >	538
base< unsigned, count_adjacent_vertices< I > >	
mln::morpho::attribute::count_adjacent_vertices< I >	826
mln::Object< count_labels >	837
mln::Meta_Accumulator< count_labels >	818
mln::accu::meta::count_labels	460
mln::Object< count_labels< L > >	837
mln::Proxy< count_labels< L > >	941
mln::Accumulator< count_labels< L > >	538
base< unsigned, count_labels< L > >	
mln::accu::count_labels< L >	438
mln::Object< count_value >	837
mln::Meta_Accumulator< count_value >	818
mln::accu::meta::count_value	461
mln::Object< count_value< V > >	837
mln::Proxy< count_value< V > >	941
mln::Accumulator< count_value< V > >	538
base< unsigned, count_value< V > >	
mln::accu::count_value< V >	439
mln::Object< couple< T, U > >	837
mln::util::couple< T, U >	1031
mln::Object< cube >	837
mln::Mesh< cube >	817
mln::Regular_Grid< cube >	944
mln::Object< cube3d >	837
mln::Window< cube3d >	1148
window_base< dpoint3d, cube3d >	
classical_window_base< dpoint3d, cube3d >	
mln::win::cube3d	1139
mln::Object< cuboid3d >	837
mln::Window< cuboid3d >	1148
window_base< dpoint3d, cuboid3d >	
classical_window_base< dpoint3d, cuboid3d >	
mln::win::cuboid3d	1140
mln::Object< cyan_t >	837
mln::Literal< cyan_t >	793
mln::literal::cyan_t	797
mln::Object< d_t >	837
mln::Function< d_t >	710
mln::Function_vv2v< d_t >	713
mln::Object< dark_gray_t >	837

mln::Literal< dark_gray_t >	793
mln::Object< dashed_line_f< I, dim > >	837
mln::Function< dashed_line_f< I, dim > >	710
mln::Function_v2v< dashed_line_f< I, dim > >	712
mln::Function_v2b< dashed_line_f< I, dim > >	711
mln::Object< decorated_image< I, D > >	837
mln::Image< decorated_image< I, D > >	757
mln::Object< depth1st_piter< T > >	837
mln::Proxy< depth1st_piter< T > >	941
mln::Site_Proxy< depth1st_piter< T > >	949
mln::Site_Iterator< depth1st_piter< T > >	947
mln::Object< depth_first_search_t >	837
mln::Browsing< depth_first_search_t >	564
graph_first_search_t< depth_first_search_t, std::stack > mln::canvas::browsing::depth_first_search_t	566
mln::Object< desc_propagation >	837
mln::Object< deviation >	837
mln::Meta_Accumulator< deviation >	818
mln::accu::stat::meta::deviation	519
mln::Object< deviation< T, S, M > >	837
mln::Proxy< deviation< T, S, M > >	941
mln::Accumulator< deviation< T, S, M > >	538
base< M, deviation< T, S, M > > mln::accu::stat::deviation< T, S, M >	507
mln::Object< diag2d >	837
mln::Window< diag2d >	1148
window_base< dpoint2d, diag2d > classical_window_base< dpoint2d, diag2d > mln::win::diag2d	1142
mln::Object< diagonal2d_t >	837
mln::Browsing< diagonal2d_t >	564
mln::canvas::browsing::diagonal2d_t	567
mln::Object< diff_abs< V > >	837
mln::Function< diff_abs< V > >	710
mln::Function_vv2v< diff_abs< V > >	713
mln::fun::vv2v::diff_abs< V >	689
mln::Object< dir_struct_elt_incr_update_t >	837
mln::Browsing< dir_struct_elt_incr_update_t >	564
mln::canvas::browsing::dir_struct_elt_incr_update_t	568
mln::Object< directional_t >	837
mln::Browsing< directional_t >	564
mln::canvas::browsing::directional_t	569
mln::Object< dist >	837
mln::Function< dist >	710
mln::Function_vv2v< dist >	713
mln::Object< dist_t >	837
mln::Function< dist_t >	710
mln::Function_vv2v< dist_t >	713

mln::Object< dn_leaf_piter< T > >	837
mln::Proxy< dn_leaf_piter< T > >	941
mln::Site_Proxy< dn_leaf_piter< T > >	949
mln::Site_Iterator< dn_leaf_piter< T > >	947
mln::Object< dn_node_piter< T > >	837
mln::Proxy< dn_node_piter< T > >	941
mln::Site_Proxy< dn_node_piter< T > >	949
mln::Site_Iterator< dn_node_piter< T > >	947
mln::Object< dn_site_piter< T > >	837
mln::Proxy< dn_site_piter< T > >	941
mln::Site_Proxy< dn_site_piter< T > >	949
mln::Site_Iterator< dn_site_piter< T > >	947
mln::Object< dpoint< G, C > >	837
mln::Gdpoint< dpoint< G, C > >	718
mln::dpoint< G, C >	639
mln::Object< dpoints_bkd_pixter< I > >	837
mln::Iterator< dpoints_bkd_pixter< I > >	781
Pixel_Iterator< dpoints_bkd_pixter< I > >	
mln::Object< dpoints_fwd_pixter< I > >	837
mln::Iterator< dpoints_fwd_pixter< I > >	781
Pixel_Iterator< dpoints_fwd_pixter< I > >	
mln::Object< dpsites_bkd_piter< V > >	837
mln::Proxy< dpsites_bkd_piter< V > >	941
mln::Site_Proxy< dpsites_bkd_piter< V > >	949
mln::Site_Iterator< dpsites_bkd_piter< V > >	947
site_iterator_base< V, dpsites_bkd_piter< V > >	
site_relative_iterator_base< V, dpsites_bkd_piter< V > >	
mln::dpsites_bkd_piter< V >	649
mln::Object< dpsites_fwd_piter< V > >	837
mln::Proxy< dpsites_fwd_piter< V > >	941
mln::Site_Proxy< dpsites_fwd_piter< V > >	949
mln::Site_Iterator< dpsites_fwd_piter< V > >	947
site_iterator_base< V, dpsites_fwd_piter< V > >	
site_relative_iterator_base< V, dpsites_fwd_piter< V > >	
mln::dpsites_fwd_piter< V >	650
mln::Object< eat >	837
mln::util::eat	1033
mln::Object< edge_bkd_iterator< G > >	837
mln::Proxy< edge_bkd_iterator< G > >	941
mln::Object< edge_fwd_iterator< G > >	837
mln::Proxy< edge_fwd_iterator< G > >	941
mln::Object< edge_image< P, V, G > >	837
mln::Image< edge_image< P, V, G > >	757
mln::Object< edge_nbh_edge_bkd_iterator< G > >	837
mln::Proxy< edge_nbh_edge_bkd_iterator< G > >	941
mln::Object< edge_nbh_edge_fwd_iterator< G > >	837
mln::Proxy< edge_nbh_edge_fwd_iterator< G > >	941
mln::Object< edge_to_color< I, V > >	837

mln::Function< edge_to_color< I, V > >	710
mln::Function_v2v< edge_to_color< I, V > >	712
mln::Object< enc< V > >	837
mln::Function< enc< V > >	710
mln::Function_v2v< enc< V > >	712
mln::Object< eq< L, R > >	837
mln::Function< eq< L, R > >	710
mln::Function_vv2b< eq< L, R > >	713
mln::fun::vv2b::eq< L, R >	684
mln::Object< extended< I > >	837
mln::Image< extended< I > >	757
mln::Object< extension_fun< I, F > >	837
mln::Image< extension_fun< I, F > >	757
mln::Object< extension_ima< I, J > >	837
mln::Image< extension_ima< I, J > >	757
mln::Object< extension_val< I > >	837
mln::Image< extension_val< I > >	757
mln::Object< f_16_to_8 >	837
mln::Function< f_16_to_8 >	710
mln::Function_v2v< f_16_to_8 >	712
mln::Object< f_box1d_t >	837
mln::Function< f_box1d_t >	710
mln::Function_v2v< f_box1d_t >	712
mln::Function_v2b< f_box1d_t >	711
mln::Object< f_box2d_t >	837
mln::Function< f_box2d_t >	710
mln::Function_v2v< f_box2d_t >	712
mln::Function_v2b< f_box2d_t >	711
mln::Object< f_box3d_t >	837
mln::Function< f_box3d_t >	710
mln::Function_v2v< f_box3d_t >	712
mln::Function_v2b< f_box3d_t >	711
mln::Object< f_hsi_to_rgb< T_rgb > >	837
mln::Function< f_hsi_to_rgb< T_rgb > >	710
mln::Function_v2v< f_hsi_to_rgb< T_rgb > >	712
mln::Object< f_hsl_to_rgb< T_rgb > >	837
mln::Function< f_hsl_to_rgb< T_rgb > >	710
mln::Function_v2v< f_hsl_to_rgb< T_rgb > >	712
mln::Object< f_rgb_to_hsi< T_hsi > >	837
mln::Function< f_rgb_to_hsi< T_hsi > >	710
mln::Function_v2v< f_rgb_to_hsi< T_hsi > >	712
mln::Object< f_rgb_to_hsl< T_hsl > >	837
mln::Function< f_rgb_to_hsl< T_hsl > >	710
mln::Function_v2v< f_rgb_to_hsl< T_hsl > >	712
mln::Object< face_bkd_iter< D > >	837
mln::Iterator< face_bkd_iter< D > >	781
complex_iterator_base< topo::face< D >, face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >	

mln::topo::face_bkd_iter< D >	993
mln::Object< face_fwd_iter< D > >	837
mln::Iterator< face_fwd_iter< D > >	781
complex_iterator_base< topo::face< D >, face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >	
mln::topo::face_fwd_iter< D >	995
mln::Object< faces_psite< N, D, P > >	837
mln::Proxy< faces_psite< N, D, P > >	941
mln::Site_Proxy< faces_psite< N, D, P > >	949
mln::Pseudo_Site< faces_psite< N, D, P > >	941
mln::Object< fibonacci_heap< P, T > >	837
mln::util::fibonacci_heap< P, T >	1038
mln::Object< flat_image< T, S > >	837
mln::Image< flat_image< T, S > >	757
mln::Object< float01 >	837
mln::Value< float01 >	1089
mln::Object< float01_f >	837
mln::Value< float01_f >	1089
mln::Object< float_off_loader >	837
mln::Object< float_off_saver >	837
mln::Object< fold< P, dir_0, dir_1, dir_2 > >	837
mln::Function< fold< P, dir_0, dir_1, dir_2 > >	710
mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > >	712
mln::Object< from_accu< A > >	837
mln::Meta_Function< from_accu< A > >	820
mln::Meta_Function_v2v< from_accu< A > >	821
mln::Object< fun_image< F, I > >	837
mln::Image< fun_image< F, I > >	757
mln::Object< function< meta::blue< value::rgb< n > > > >	837
mln::Function< function< meta::blue< value::rgb< n > > > >	710
mln::Function_v2v< function< meta::blue< value::rgb< n > > > >	712
mln::Object< function< meta::first< util::couple< T, U > > > >	837
mln::Function< function< meta::first< util::couple< T, U > > > >	710
mln::Function_v2v< function< meta::first< util::couple< T, U > > > >	712
mln::Object< function< meta::green< value::rgb< n > > > >	837
mln::Function< function< meta::green< value::rgb< n > > > >	710
mln::Function_v2v< function< meta::green< value::rgb< n > > > >	712
mln::Object< function< meta::red< value::rgb< n > > > >	837
mln::Function< function< meta::red< value::rgb< n > > > >	710
mln::Function_v2v< function< meta::red< value::rgb< n > > > >	712
mln::Object< function< meta::second< util::couple< T, U > > > >	837
mln::Function< function< meta::second< util::couple< T, U > > > >	710
mln::Function_v2v< function< meta::second< util::couple< T, U > > > >	712
mln::Object< function< meta::to_enc< T > > >	837
mln::Function< function< meta::to_enc< T > > >	710
mln::Function_v2v< function< meta::to_enc< T > > >	712
mln::Object< fwd_pixter1d< I > >	837
mln::Iterator< fwd_pixter1d< I > >	781

Pixel_Iterator< fwd_pixter1d< I > >	
mln::Object< fwd_pixter2d< I > >	837
mln::Iterator< fwd_pixter2d< I > >	781
Pixel_Iterator< fwd_pixter2d< I > >	
mln::Object< fwd_pixter3d< I > >	837
mln::Iterator< fwd_pixter3d< I > >	781
Pixel_Iterator< fwd_pixter3d< I > >	
mln::Object< fwd_t >	837
mln::Browsing< fwd_t >	564
mln::canvas::browsing::fwd_t	571
mln::Object< ge< L, R > >	837
mln::Function< ge< L, R > >	710
mln::Function_vv2b< ge< L, R > >	713
mln::fun::vv2b::ge< L, R >	684
mln::Object< graph >	837
mln::Graph< graph >	726
graph_base< graph >	
mln::util::graph	1042
mln::Object< graph_elt_mixed_window< G, S, S2 > >	837
mln::Window< graph_elt_mixed_window< G, S, S2 > >	1148
mln::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > >	746
mln::graph_elt_mixed_window< G, S, S2 >	729
mln::Object< graph_elt_window< G, S > >	837
mln::Window< graph_elt_window< G, S > >	1148
mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > >	746
mln::graph_elt_window< G, S >	737
mln::Object< graph_elt_window_if< G, S, I > >	837
mln::Window< graph_elt_window_if< G, S, I > >	1148
mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >	746
mln::graph_elt_window_if< G, S, I >	741
mln::Object< graph_window_if_piter< S, W, I > >	837
mln::Proxy< graph_window_if_piter< S, W, I > >	941
mln::Site_Proxy< graph_window_if_piter< S, W, I > >	949
mln::Site_Iterator< graph_window_if_piter< S, W, I > >	947
site_iterator_base< W, graph_window_if_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >	
mln::graph_window_if_piter< S, W, I >	748
mln::Object< graph_window_piter< S, W, I > >	837
mln::Proxy< graph_window_piter< S, W, I > >	941
mln::Site_Proxy< graph_window_piter< S, W, I > >	949
mln::Site_Iterator< graph_window_piter< S, W, I > >	947
site_iterator_base< W, graph_window_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >	
mln::graph_window_piter< S, W, I >	750
mln::Object< gray_f >	837
mln::Value< gray_f >	1089
mln::Object< graylevel< n > >	837
mln::Value< graylevel< n > >	1089

mln::Object< graylevel_f >	837
mln::Value< graylevel_f >	1089
mln::Object< green >	837
mln::Meta_Function< green >	820
mln::Meta_Function_v2v< green >	821
mln::Object< green_t >	837
mln::Literal< green_t >	793
mln::literal::green_t	798
mln::Object< gt< L, R > >	837
mln::Function< gt< L, R > >	710
mln::Function_vv2b< gt< L, R > >	713
mln::fun::vv2b::gt< L, R >	685
mln::Object< has< I > >	837
mln::Function< has< I > >	710
mln::Function_v2v< has< I > >	712
mln::Function_v2b< has< I > >	711
mln::Object< height >	837
mln::Meta_Accumulator< height >	818
mln::accu::meta::shape::height	479
mln::Object< height< I > >	837
mln::Proxy< height< I > >	941
mln::Accumulator< height< I > >	538
base< unsigned, height< I > >	
mln::accu::shape::height< I >	501
mln::morpho::attribute::height< I >	827
mln::Object< hexa >	837
mln::Mesh< hexa >	817
mln::Regular_Grid< hexa >	944
mln::Object< hexa< I > >	837
mln::Image< hexa< I > >	757
mln::Object< hexa< image2d< V > > >	837
mln::Image< hexa< image2d< V > > >	757
mln::Object< histo >	837
mln::Meta_Accumulator< histo >	818
mln::accu::meta::histo	462
mln::Object< histo3d_rgb >	837
mln::Meta_Accumulator< histo3d_rgb >	818
mln::Object< histo3d_rgb< V > >	837
mln::Proxy< histo3d_rgb< V > >	941
mln::Accumulator< histo3d_rgb< V > >	538
base< image3d< unsigned >, histo3d_rgb< V > >	
mln::accu::stat::histo3d_rgb< V >	509
mln::Object< histo< V > >	837
mln::Proxy< histo< V > >	941
mln::Accumulator< histo< V > >	538
base< const std::vector< unsigned > &, histo< V > >	
mln::accu::histo< V >	441
mln::Object< hyper_directional_t >	837

mln::Browsing< hyper_directional_t >	564
mln::canvas::browsing::hyper_directional_t	572
mln::Object< id2element< G, Elt > >	837
mln::Function< id2element< G, Elt > >	710
mln::Function_v2v< id2element< G, Elt > >	712
mln::Object< identity_t >	837
mln::Literal< identity_t >	793
mln::literal::identity_t	799
mln::Object< ignore >	837
mln::util::ignore	1050
mln::Object< image1d< T > >	837
mln::Image< image1d< T > >	757
mln::Object< image2d< T > >	837
mln::Image< image2d< T > >	757
mln::Object< image3d< T > >	837
mln::Image< image3d< T > >	757
mln::Object< image< F, S > >	837
mln::Image< image< F, S > >	757
mln::Object< image_if< I, F > >	837
mln::Image< image_if< I, F > >	757
mln::Object< implies< L, R > >	837
mln::Function< implies< L, R > >	710
mln::Function_vv2b< implies< L, R > >	713
mln::fun::vv2b::implies< L, R >	686
mln::Object< index_of_value< bool > >	837
mln::Function< index_of_value< bool > >	710
mln::Function_v2v< index_of_value< bool > >	712
mln::Object< index_of_value< T > >	837
mln::Function< index_of_value< T > >	710
mln::Function_v2v< index_of_value< T > >	712
mln::Object< inf >	837
mln::Meta_Accumulator< inf >	818
mln::accu::meta::math::inf	470
mln::Meta_Function< inf >	820
mln::Meta_Function_vv2v< inf >	821
mln::Object< inf< T > >	837
mln::Proxy< inf< T > >	941
mln::Accumulator< inf< T > >	538
base< const T &, inf< T > >	
mln::accu::math::inf< T >	453
mln::Object< int_s< n > >	837
mln::Object< int_u8_off_saver >	837
mln::Object< int_u< n > >	837
mln::Object< int_u_sat< n > >	837
mln::Value< int_u_sat< n > >	1089
mln::Object< interpolated< I, F > >	837
mln::Image< interpolated< I, F > >	757
mln::Object< iota >	837

mln::Function< iota >	710
mln::Function_v2v< iota >	712
mln::Object< is_dot >	837
mln::Function< is_dot >	710
mln::Function_v2v< is_dot >	712
mln::Function_v2b< is_dot >	711
mln::Object< is_edge >	837
mln::Function< is_edge >	710
mln::Function_v2v< is_edge >	712
mln::Function_v2b< is_edge >	711
mln::Object< is_n_face< N > >	837
mln::Function< is_n_face< N > >	710
mln::Function_v2v< is_n_face< N > >	712
mln::Function_v2b< is_n_face< N > >	711
mln::topo::is_n_face< N >	996
mln::Object< is_pixel >	837
mln::Function< is_pixel >	710
mln::Function_v2v< is_pixel >	712
mln::Function_v2b< is_pixel >	711
mln::Object< is_row_odd >	837
mln::Function< is_row_odd >	710
mln::Function_v2v< is_row_odd >	712
mln::Function_v2b< is_row_odd >	711
mln::Object< is_separator >	837
mln::Function< is_separator >	710
mln::Function_v2v< is_separator >	712
mln::Function_v2b< is_separator >	711
mln::world::inter_pixel::is_separator	1154
mln::Object< is_simple_cell< I > >	837
mln::Function< is_simple_cell< I > >	710
mln::Function_v2v< is_simple_cell< I > >	712
mln::Function_v2b< is_simple_cell< I > >	711
mln::topo::is_simple_cell< I >	998
mln::Object< ithcomp >	837
mln::Meta_Function< ithcomp >	820
mln::Meta_Function_vv2v< ithcomp >	821
mln::Object< keep_specific_colors >	837
mln::Function< keep_specific_colors >	710
mln::Function_v2v< keep_specific_colors >	712
mln::Function_v2b< keep_specific_colors >	711
mln::Object< l1 >	837
mln::Meta_Function< l1 >	820
mln::Meta_Function_v2v< l1 >	821
mln::Object< l1_norm< V > >	837
mln::Function< l1_norm< V > >	710
mln::Function_v2v< l1_norm< V > >	712
mln::Object< l1_norm< V, R > >	837
mln::Function< l1_norm< V, R > >	710

mln::Function_v2v< l1_norm< V, R > >	712
mln::fun::v2v::l1_norm< V, R >	673
mln::fun::v2w_w2v::l1_norm< V, R >	680
mln::Object< l2 >	837
mln::Meta_Function< l2 >	820
mln::Meta_Function_v2v< l2 >	821
mln::Object< l2_norm< V, R > >	837
mln::Function< l2_norm< V, R > >	710
mln::Function_v2v< l2_norm< V, R > >	712
mln::fun::v2v::l2_norm< V, R >	674
mln::fun::v2w_w2v::l2_norm< V, R >	682
mln::Object< label< n > >	837
mln::Value< label< n > >	1089
mln::Object< label_used >	837
mln::Meta_Accumulator< label_used >	818
mln::accu::meta::label_used	463
mln::Object< label_used< L > >	837
mln::Proxy< label_used< L > >	941
mln::Accumulator< label_used< L > >	538
base< const fun::i2v::array< bool > &, label_used< L > >	442
mln::accu::label_used< L >	442
mln::Object< labeled_image< I > >	837
mln::Image< labeled_image< I > >	757
mln::Object< land >	837
mln::Meta_Accumulator< land >	818
mln::accu::meta::logic::land	464
mln::Proxy< land >	941
mln::Accumulator< land >	538
base< bool, land >	444
mln::accu::logic::land	444
mln::Object< land< L, R > >	837
mln::Function< land< L, R > >	710
mln::Function_vv2v< land< L, R > >	713
mln::fun::vv2v::land< L, R >	690
mln::Object< land_basic >	837
mln::Meta_Accumulator< land_basic >	818
mln::accu::meta::logic::land_basic	465
mln::Proxy< land_basic >	941
mln::Accumulator< land_basic >	538
base< bool, land_basic >	445
mln::accu::logic::land_basic	445
mln::Object< land_not< L, R > >	837
mln::Function< land_not< L, R > >	710
mln::Function_vv2v< land_not< L, R > >	713
mln::fun::vv2v::land_not< L, R >	691
mln::Object< lazy_image< I, F, B > >	837
mln::Image< lazy_image< I, F, B > >	757
mln::Object< le< L, R > >	837
mln::Function< le< L, R > >	710

mln::Function_vv2b< le< L, R > >	713
mln::fun::vv2b::le< L, R >	687
mln::Object< light_gray_t >	837
mln::Literal< light_gray_t >	793
mln::literal::light_gray_t	800
mln::Object< lime_t >	837
mln::Literal< lime_t >	793
mln::literal::lime_t	801
mln::Object< line< M, i, C > >	837
mln::Window< line< M, i, C > >	1148
window_base< dpoint< M, C >, line< M, i, C > >	
classical_window_base< dpoint< M, C >, line< M, i, C > >	
mln::win::line< M, i, C >	1143
mln::Object< line_graph< G > >	837
mln::Graph< line_graph< G > >	726
graph_base< line_graph< G > >	
mln::util::line_graph< G >	1051
mln::Object< linear< V, T, R > >	837
mln::Function< linear< V, T, R > >	710
mln::Function_v2v< linear< V, T, R > >	712
mln::fun::v2v::linear< V, T, R >	675
mln::Object< linear_sat< V, T, R > >	837
mln::Function< linear_sat< V, T, R > >	710
mln::Function_v2v< linear_sat< V, T, R > >	712
mln::Object< linfty >	837
mln::Meta_Function< linfty >	820
mln::Meta_Function_v2v< linfty >	821
mln::Object< linfty_norm< V, R > >	837
mln::Function< linfty_norm< V, R > >	710
mln::Function_v2v< linfty_norm< V, R > >	712
mln::fun::v2v::linfty_norm< V, R >	676
mln::fun::v2w_w2v::linfty_norm< V, R >	683
mln::Object< lnot< V > >	837
mln::Function< lnot< V > >	710
mln::Function_v2v< lnot< V > >	712
mln::Function_v2b< lnot< V > >	711
mln::fun::v2b::lnot< V >	670
mln::Object< lor >	837
mln::Meta_Accumulator< lor >	818
mln::accu::meta::logic::lor	466
mln::Proxy< lor >	941
mln::Accumulator< lor >	538
base< bool, lor >	
mln::accu::logic::lor	447
mln::Object< lor< L, R > >	837
mln::Function< lor< L, R > >	710
mln::Function_vv2v< lor< L, R > >	713
mln::fun::vv2v::lor< L, R >	692
mln::Object< lor_basic >	837

mln::Meta_Accumulator< lor_basic >	818
mln::accu::meta::logic::lor_basic	467
mln::Proxy< lor_basic >	941
mln::Accumulator< lor_basic >	538
base< bool, lor_basic >	
mln::accu::logic::lor_basic	448
mln::Object< lt< L, R > >	837
mln::Function< lt< L, R > >	710
mln::Function_vv2b< lt< L, R > >	713
mln::fun::vv2b::lt< L, R >	688
mln::Object< lut_vec< S, T > >	837
Value_Set< lut_vec< S, T > >	
mln::value::lut_vec< S, T >	1111
mln::Object< lxor< L, R > >	837
mln::Function< lxor< L, R > >	710
mln::Function_vv2v< lxor< L, R > >	713
mln::fun::vv2v::lxor< L, R >	693
mln::Object< magenta_t >	837
mln::Literal< magenta_t >	793
mln::literal::magenta_t	802
mln::Object< mahalanobis< V > >	837
mln::Function< mahalanobis< V > >	710
mln::Function_v2v< mahalanobis< V > >	712
mln::Object< maj_h >	837
mln::Meta_Accumulator< maj_h >	818
mln::accu::meta::maj_h	468
mln::Object< maj_h< T > >	837
mln::Proxy< maj_h< T > >	941
mln::Accumulator< maj_h< T > >	538
base< const T &, maj_h< T > >	
mln::accu::maj_h< T >	450
mln::Object< mat< n, m, T > >	837
mln::Object< max >	837
mln::Meta_Accumulator< max >	818
mln::accu::meta::stat::max	481
mln::Object< max< T > >	837
mln::Proxy< max< T > >	941
mln::Accumulator< max< T > >	538
base< const T &, max< T > >	
mln::accu::stat::max< T >	511
mln::Object< max< V > >	837
mln::Function< max< V > >	710
mln::Function_vv2v< max< V > >	713
mln::fun::vv2v::max< V >	694
mln::Object< max_h >	837
mln::Meta_Accumulator< max_h >	818
mln::accu::meta::stat::max_h	482
mln::Object< max_h< V > >	837
mln::Proxy< max_h< V > >	941

mln::Accumulator< max_h< V > >	538
base< const V &, max_h< V > >	
mln::accu::stat::max_h< V >	513
mln::Object< max_site >	837
mln::Meta_Accumulator< max_site >	818
mln::accu::meta::max_site	473
mln::Object< max_site< I > >	837
mln::Proxy< max_site< I > >	941
mln::Accumulator< max_site< I > >	538
base< I::psite, max_site< I > >	
mln::accu::max_site< I >	457
mln::Object< max_t >	837
mln::Literal< max_t >	793
mln::literal::max_t	803
mln::Object< mean >	837
mln::Meta_Accumulator< mean >	818
mln::accu::meta::stat::mean	483
mln::Meta_Function< mean >	820
mln::Meta_Function_v2v< mean >	821
mln::Object< mean< T, S, M > >	837
mln::Proxy< mean< T, S, M > >	941
mln::Accumulator< mean< T, S, M > >	538
base< M, mean< T, S, M > >	
mln::accu::stat::mean< T, S, M >	514
mln::Object< median_alt< S > >	837
mln::Proxy< median_alt< S > >	941
mln::Accumulator< median_alt< S > >	538
base< const S::value &, median_alt< S > >	
mln::accu::stat::median_alt< S >	516
mln::Object< median_alt< T > >	837
mln::Meta_Accumulator< median_alt< T > >	818
mln::accu::meta::stat::median_alt< T >	484
mln::Object< median_alt< value::set< T > > >	837
mln::Proxy< median_alt< value::set< T > > >	941
mln::Accumulator< median_alt< value::set< T > > >	538
base< const value::set< T >::value &, median_alt< value::set< T > > >	
mln::accu::stat::median_alt< value::set< T > > >	516
mln::Object< median_h >	837
mln::Meta_Accumulator< median_h >	818
mln::accu::meta::stat::median_h	485
mln::Object< median_h< V > >	837
mln::Proxy< median_h< V > >	941
mln::Accumulator< median_h< V > >	538
base< const V &, median_h< V > >	
mln::accu::stat::median_h< V >	518
mln::Object< medium_gray_t >	837
mln::Literal< medium_gray_t >	793
mln::Object< min >	837
mln::Meta_Accumulator< min >	818

mln::accu::meta::stat::min	486
mln::Object< min< L, R > >	837
mln::Function< min< L, R > >	710
mln::Function_vv2v< min< L, R > >	713
mln::fun::vv2v::min< L, R >	695
mln::Object< min< T > >	837
mln::Proxy< min< T > >	941
mln::Accumulator< min< T > >	538
base< const T &, min< T > >	
mln::accu::stat::min< T >	520
mln::Object< min_h >	837
mln::Meta_Accumulator< min_h >	818
mln::accu::meta::stat::min_h	487
mln::Object< min_h< V > >	837
mln::Proxy< min_h< V > >	941
mln::Accumulator< min_h< V > >	538
base< const V &, min_h< V > >	
mln::accu::stat::min_h< V >	522
mln::Object< min_t >	837
mln::Literal< min_t >	793
mln::literal::min_t	804
mln::Object< mirror< B > >	837
mln::Function< mirror< B > >	710
mln::Function_v2v< mirror< B > >	712
mln::Object< mixed_neighb< W > >	837
mln::Neighborhood< mixed_neighb< W > >	836
mln::Object< mln::util::set< T > >	837
mln::util::set< T >	1063
mln::Object< multi_site< P > >	837
mln::Object< multiple< W, F > >	837
mln::Window< multiple< W, F > >	1148
window_base< W::dpsite, multiple< W, F > >	
mln::win::multiple< W, F >	1145
mln::Object< multiple_qiter< W, F > >	837
mln::Proxy< multiple_qiter< W, F > >	941
mln::Site_Proxy< multiple_qiter< W, F > >	949
mln::Site_Iterator< multiple_qiter< W, F > >	947
mln::Object< multiple_size< n, W, F > >	837
mln::Window< multiple_size< n, W, F > >	1148
window_base< W::dpsite, multiple_size< n, W, F > >	
mln::win::multiple_size< n, W, F >	1145
mln::Object< multiple_size_qiter< n, W, F > >	837
mln::Proxy< multiple_size_qiter< n, W, F > >	941
mln::Site_Proxy< multiple_size_qiter< n, W, F > >	949
mln::Site_Iterator< multiple_size_qiter< n, W, F > >	947
mln::Object< my_box2d >	837
mln::Function< my_box2d >	710
mln::Function_v2v< my_box2d >	712

mln::Function_v2b< my_box2d >	711
mln::Object< my_ext >	837
mln::Function< my_ext >	710
mln::Function_v2v< my_ext >	712
mln::Object< my_fun< G > >	837
mln::Function< my_fun< G > >	710
mln::Object< my_image2d< T > >	837
mln::Image< my_image2d< T > >	757
mln::Object< my_values_t >	837
mln::Function< my_values_t >	710
mln::Function_v2v< my_values_t >	712
mln::Object< myfun >	837
mln::Function< myfun >	710
mln::Function_vv2v< myfun >	713
mln::Object< mysqrt >	837
mln::Function< mysqrt >	710
mln::Function_v2v< mysqrt >	712
mln::Object< n_face_bkd_iter< D > >	837
mln::Iterator< n_face_bkd_iter< D > >	781
complex_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
mln::topo::n_face_bkd_iter< D >	1005
mln::Object< n_face_fwd_iter< D > >	837
mln::Iterator< n_face_fwd_iter< D > >	781
complex_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
mln::topo::n_face_fwd_iter< D >	1006
mln::Object< neighb< graph_elt_mixed_window< G, S, S2 > > >	837
mln::Neighborhood< neighb< graph_elt_mixed_window< G, S, S2 > > >	836
mln::Object< neighb< graph_elt_window< G, S > > >	837
mln::Neighborhood< neighb< graph_elt_window< G, S > > >	836
mln::Object< neighb< graph_elt_window_if< G, S, I > > >	837
mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > >	836
mln::Object< neighb< W > >	837
mln::Neighborhood< neighb< W > >	836
mln::Object< neighb_bkd_niter< W > >	837
mln::Proxy< neighb_bkd_niter< W > >	941
mln::Site_Proxy< neighb_bkd_niter< W > >	949
mln::Site_Iterator< neighb_bkd_niter< W > >	947
mln::Object< neighb_fwd_niter< W > >	837
mln::Proxy< neighb_fwd_niter< W > >	941
mln::Site_Proxy< neighb_fwd_niter< W > >	949
mln::Site_Iterator< neighb_fwd_niter< W > >	947
mln::Object< nil >	837
mln::Meta_Accumulator< nil >	818
mln::accu::meta::nil	474
mln::util::nil	1056
mln::Object< nil< T > >	837

mln::Proxy< nil< T > >	941
mln::Accumulator< nil< T > >	538
base< util::ignore, nil< T > >	
mln::accu::nil< T >	492
mln::Object< not_to_remove >	837
mln::Function< not_to_remove >	710
mln::Function_v2v< not_to_remove >	712
mln::Function_v2b< not_to_remove >	711
mln::Object< object_id< Tag, V > >	837
mln::Value< object_id< Tag, V > >	1089
mln::Object< octagon2d >	837
mln::Window< octagon2d >	1148
window_base< dpoint2d, octagon2d >	
classical_window_base< dpoint2d, octagon2d >	
mln::win::octagon2d	1145
mln::Object< olive_t >	837
mln::Literal< olive_t >	793
mln::literal::olive_t	805
mln::Object< one_t >	837
mln::Literal< one_t >	793
mln::literal::one_t	806
mln::Object< orange_t >	837
mln::Literal< orange_t >	793
mln::literal::orange_t	807
mln::Object< ord_pair< T > >	837
mln::util::ord_pair< T >	1059
mln::Object< origin_t >	837
mln::Literal< origin_t >	793
mln::literal::origin_t	808
mln::Object< P >	837
Point_Site< P >	
mln::Point< P >	932
mln::Object< p2p_image< I, F > >	837
mln::Image< p2p_image< I, F > >	757
mln::Object< p< A > >	837
mln::Proxy< p< A > >	941
mln::Accumulator< p< A > >	538
base< const A::result &, p< A > >	
mln::accu::p< A >	494
mln::Object< p< mA > >	837
mln::Meta_Accumulator< p< mA > >	818
mln::accu::meta::p< mA >	475
mln::Object< p_array< P > >	837
mln::Site_Set< p_array< P > >	950
site_set_base_< P, p_array< P > >	
mln::p_array< P >	839
mln::Object< p_centered< W > >	837
mln::Site_Set< p_centered< W > >	950

site_set_base_< W::psite, p_centered< W > >	
mln::p_centered< W >	845
mln::Object< p_centered_piter< W > >	837
mln::Proxy< p_centered_piter< W > >	941
mln::Site_Proxy< p_centered_piter< W > >	949
mln::Site_Iterator< p_centered_piter< W > >	947
mln::Object< p_complex< D, G > >	837
mln::Site_Set< p_complex< D, G > >	950
site_set_base_< complex_psite< D, G >, p_complex< D, G > >	
mln::p_complex< D, G >	848
mln::Object< p_double_piter< S, I1, I2 > >	837
mln::Proxy< p_double_piter< S, I1, I2 > >	941
mln::Site_Proxy< p_double_piter< S, I1, I2 > >	949
mln::Site_Iterator< p_double_piter< S, I1, I2 > >	947
mln::Object< p_double_psite< S, Sp > >	837
mln::Proxy< p_double_psite< S, Sp > >	941
mln::Site_Proxy< p_double_psite< S, Sp > >	949
mln::Pseudo_Site< p_double_psite< S, Sp > >	941
mln::Object< p_edges< G, F > >	837
mln::Site_Set< p_edges< G, F > >	950
site_set_base_< F::result, p_edges< G, F > >	
mln::p_edges< G, F >	852
mln::Object< p_edges_psite< G, F > >	837
mln::Proxy< p_edges_psite< G, F > >	941
mln::Site_Proxy< p_edges_psite< G, F > >	949
mln::Pseudo_Site< p_edges_psite< G, F > >	941
mln::Object< p_faces< N, D, P > >	837
mln::Site_Set< p_faces< N, D, P > >	950
site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >	
mln::p_faces< N, D, P >	858
mln::Object< p_graph_piter< S, I > >	837
mln::Proxy< p_graph_piter< S, I > >	941
mln::Site_Proxy< p_graph_piter< S, I > >	949
mln::Site_Iterator< p_graph_piter< S, I > >	947
site_iterator_base< S, p_graph_piter< S, I > >	
site_set_iterator_base< S, p_graph_piter< S, I > >	
mln::p_graph_piter< S, I >	861
mln::Object< p_if< S, F > >	837
mln::Site_Set< p_if< S, F > >	950
site_set_base_< S::psite, p_if< S, F > >	
mln::p_if< S, F >	862
mln::Object< p_image< I > >	837
mln::Site_Set< p_image< I > >	950
site_set_base_< I::psite, p_image< I > >	
mln::p_image< I >	866
mln::Object< p_indexed_bkd_piter< S > >	837
mln::Proxy< p_indexed_bkd_piter< S > >	941
mln::Site_Proxy< p_indexed_bkd_piter< S > >	949
mln::Site_Iterator< p_indexed_bkd_piter< S > >	947

site_iterator_base< S, p_indexed_bkd_piter< S > >	
site_set_iterator_base< S, p_indexed_bkd_piter< S > >	
mln::p_indexed_bkd_piter< S >	870
mln::Object< p_indexed_fwd_piter< S > >	837
mln::Proxy< p_indexed_fwd_piter< S > >	941
mln::Site_Proxy< p_indexed_fwd_piter< S > >	949
mln::Site_Iterator< p_indexed_fwd_piter< S > >	947
site_iterator_base< S, p_indexed_fwd_piter< S > >	
site_set_iterator_base< S, p_indexed_fwd_piter< S > >	
mln::p_indexed_fwd_piter< S >	871
mln::Object< p_indexed_psite< S > >	837
mln::Proxy< p_indexed_psite< S > >	941
mln::Site_Proxy< p_indexed_psite< S > >	949
mln::Pseudo_Site< p_indexed_psite< S > >	941
pseudo_site_base_< const S::element &, p_indexed_psite< S > >	
mln::p_indexed_psite< S >	873
mln::Object< p_key< K, P > >	837
mln::Site_Set< p_key< K, P > >	950
site_set_base_< P, p_key< K, P > >	
mln::p_key< K, P >	873
mln::Object< p_line2d >	837
mln::Site_Set< p_line2d >	950
site_set_base_< point2d, p_line2d >	
mln::p_line2d	878
mln::Object< p_mutable_array_of< S > >	837
mln::Site_Set< p_mutable_array_of< S > >	950
site_set_base_< S::site, p_mutable_array_of< S > >	
mln::p_mutable_array_of< S >	882
mln::Object< p_n_faces_bkd_piter< D, G > >	837
mln::Proxy< p_n_faces_bkd_piter< D, G > >	941
mln::Site_Proxy< p_n_faces_bkd_piter< D, G > >	949
mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >	947
site_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >	
mln::p_n_faces_bkd_piter< D, G >	886
mln::Object< p_n_faces_fwd_piter< D, G > >	837
mln::Proxy< p_n_faces_fwd_piter< D, G > >	941
mln::Site_Proxy< p_n_faces_fwd_piter< D, G > >	949
mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >	947
site_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >	
mln::p_n_faces_fwd_piter< D, G >	887
mln::Object< p_priority< P, Q > >	837
mln::Site_Set< p_priority< P, Q > >	950
site_set_base_< Q::site, p_priority< P, Q > >	
mln::p_priority< P, Q >	889

mln::Object< p_queue< P > >	837
mln::Site_Set< p_queue< P > >	950
site_set_base_< P, p_queue< P > >	
mln::p_queue< P >	894
mln::Object< p_queue_fast< P > >	837
mln::Site_Set< p_queue_fast< P > >	950
site_set_base_< P, p_queue_fast< P > >	
mln::p_queue_fast< P >	899
mln::Object< p_run< P > >	837
mln::Site_Set< p_run< P > >	950
site_set_base_< P, p_run< P > >	
mln::p_run< P >	904
mln::Object< p_run_psite< P > >	837
mln::Proxy< p_run_psite< P > >	941
mln::Site_Proxy< p_run_psite< P > >	949
mln::Pseudo_Site< p_run_psite< P > >	941
mln::Object< p_set< P > >	837
mln::Site_Set< p_set< P > >	950
site_set_base_< P, p_set< P > >	
mln::p_set< P >	909
mln::Object< p_set_of< S > >	837
mln::Site_Set< p_set_of< S > >	950
mln::Object< p_transformed< S, F > >	837
mln::Site_Set< p_transformed< S, F > >	950
site_set_base_< S::psite, p_transformed< S, F > >	
mln::p_transformed< S, F >	913
mln::Object< p_transformed_piter< Pi, S, F > >	837
mln::Proxy< p_transformed_piter< Pi, S, F > >	941
mln::Site_Proxy< p_transformed_piter< Pi, S, F > >	949
mln::Site_Iterator< p_transformed_piter< Pi, S, F > >	947
mln::Object< p_vaccess< V, S > >	837
mln::Site_Set< p_vaccess< V, S > >	950
site_set_base_< S::site, p_vaccess< V, S > >	
mln::p_vaccess< V, S >	918
mln::Object< p_vertices< G, F > >	837
mln::Site_Set< p_vertices< G, F > >	950
site_set_base_< F::result, p_vertices< G, F > >	
mln::p_vertices< G, F >	922
mln::Object< p_vertices_psite< G, F > >	837
mln::Proxy< p_vertices_psite< G, F > >	941
mln::Site_Proxy< p_vertices_psite< G, F > >	949
mln::Pseudo_Site< p_vertices_psite< G, F > >	941
mln::Object< pair< A1, A2 > >	837
mln::Meta_Accumulator< pair< A1, A2 > >	818
mln::accu::meta::pair< A1, A2 >	476
mln::Object< pair< A1, A2, T > >	837
mln::Proxy< pair< A1, A2, T > >	941
mln::Accumulator< pair< A1, A2, T > >	538
base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >	

mln::accu::pair< A1, A2, T >	495
mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > >	837
mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > >	941
mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > >	538
mln::Object< pink_t >	837
mln::Literal< pink_t >	793
mln::literal::pink_t	809
mln::Object< pixel< I > >	837
mln::pixel< I >	928
mln::Object< plain< I > >	837
mln::Image< plain< I > >	757
mln::Object< point< G, C > >	837
mln::Site< point< G, C > >	946
mln::Gpoint< point< G, C > >	722
mln::point< G, C >	934
mln::Object< point_from_value< T > >	837
mln::Function< point_from_value< T > >	710
mln::Function_v2v< point_from_value< T > >	712
mln::Object< projection< P, dir > >	837
mln::Function< projection< P, dir > >	710
mln::Function_v2v< projection< P, dir > >	712
mln::Object< proxy< I > >	837
mln::Proxy< proxy< I > >	941
mln::value::proxy< I >	1114
mln::Object< purple_t >	837
mln::Literal< purple_t >	793
mln::literal::purple_t	810
mln::Object< qrde >	837
mln::Function< qrde >	710
mln::Function_v2v< qrde >	712
mln::Object< qt_rgb_to_int_u< n > >	837
mln::Function< qt_rgb_to_int_u< n > >	710
mln::Function_v2v< qt_rgb_to_int_u< n > >	712
mln::Object< quat >	837
mln::Value< quat >	1089
mln::Object< rank >	837
mln::Meta_Accumulator< rank >	818
mln::accu::meta::stat::rank	488
mln::Object< rank< bool > >	837
mln::Proxy< rank< bool > >	941
mln::Accumulator< rank< bool > >	538
base< bool, rank< bool > >	
mln::accu::stat::rank< bool >	528
mln::Object< rank< T > >	837
mln::Proxy< rank< T > >	941
mln::Accumulator< rank< T > >	538
base< const T &, rank< T > >	
mln::accu::stat::rank< T >	526

mln::Object< rank_high_quant >	837
mln::Meta_Accumulator< rank_high_quant >	818
mln::accu::meta::stat::rank_high_quant	489
mln::Object< rank_high_quant< T > >	837
mln::Proxy< rank_high_quant< T > >	941
mln::Accumulator< rank_high_quant< T > >	538
base< const T &, rank_high_quant< T > >	
mln::accu::stat::rank_high_quant< T >	529
mln::Object< rectangle2d >	837
mln::Window< rectangle2d >	1148
window_base< dpoint2d, rectangle2d >	
classical_window_base< dpoint2d, rectangle2d >	
mln::win::rectangle2d	1147
mln::Object< rectangularity< P > >	837
mln::Proxy< rectangularity< P > >	941
mln::Accumulator< rectangularity< P > >	538
mln::Object< red >	837
mln::Meta_Function< red >	820
mln::Meta_Function_v2v< red >	821
mln::Object< red_t >	837
mln::Literal< red_t >	793
mln::literal::red_t	811
mln::Object< ref_data >	837
mln::Function< ref_data >	710
mln::Function_v2v< ref_data >	712
mln::Object< rgb32 >	837
mln::Value< rgb32 >	1089
mln::Object< rgb8_off_loader >	837
mln::Object< rgb8_off_saver >	837
mln::Object< rgb8_to_rgbn< n > >	837
mln::Function< rgb8_to_rgbn< n > >	710
mln::Function_v2v< rgb8_to_rgbn< n > >	712
mln::fun::v2v::rgb8_to_rgbn< n >	677
mln::Object< rgb< n > >	837
mln::Value< rgb< n > >	1089
mln::Object< rgb_to_dist< T, n > >	837
mln::Function< rgb_to_dist< T, n > >	710
mln::Function_v2v< rgb_to_dist< T, n > >	712
mln::Object< rgb_to_int_u< n > >	837
mln::Function< rgb_to_int_u< n > >	710
mln::Function_v2v< rgb_to_int_u< n > >	712
mln::Object< rgb_to_luma< T_luma > >	837
mln::Function< rgb_to_luma< T_luma > >	710
mln::Function_v2v< rgb_to_luma< T_luma > >	712
mln::Object< rgbn_to_lbl8< n > >	837
mln::Function< rgbn_to_lbl8< n > >	710
mln::Function_v2v< rgbn_to_lbl8< n > >	712
mln::Object< rms >	837

mln::Meta_Accumulator< rms >	818
mln::accu::meta::rms	477
mln::Object< rms< T, V > >	837
mln::Proxy< rms< T, V > >	941
mln::Accumulator< rms< T, V > >	538
base< V, rms< T, V > >	
mln::accu::rms< T, V >	498
mln::Object< rotation< n, C > >	837
mln::Function< rotation< n, C > >	710
mln::Function_v2v< rotation< n, C > >	712
mln::fun::x2x::rotation< n, C >	701
mln::Object< round< R > >	837
mln::Function< round< R > >	710
mln::Function_v2v< round< R > >	712
mln::Object< row >	837
mln::Meta_Function< row >	820
mln::Meta_Function_v2v< row >	821
mln::Object< safe_image< I > >	837
mln::Image< safe_image< I > >	757
mln::Object< saturate< V > >	837
mln::Function< saturate< V > >	710
mln::Function_v2v< saturate< V > >	712
mln::Object< saturate_rgb8 >	837
mln::Function< saturate_rgb8 >	710
mln::Function_v2v< saturate_rgb8 >	712
mln::Object< scomp< ith > >	837
mln::Meta_Function< scomp< ith > >	820
mln::Meta_Function_v2v< scomp< ith > >	821
mln::Object< set_bkd_iter< T > >	837
mln::Proxy< set_bkd_iter< T > >	941
mln::Object< set_fwd_iter< T > >	837
mln::Proxy< set_fwd_iter< T > >	941
mln::Object< sharpness< I > >	837
mln::Proxy< sharpness< I > >	941
mln::Accumulator< sharpness< I > >	538
base< double, sharpness< I > >	
mln::morpho::attribute::sharpness< I >	829
mln::Object< shell< F, I > >	837
mln::Proxy< shell< F, I > >	941
mln::Object< sign >	837
mln::Value< sign >	1089
mln::Object< site_pair< P > >	837
mln::util::site_pair< P >	1069
mln::Object< sli >	837
mln::Meta_Function< sli >	820
mln::Meta_Function_v2v< sli >	821
mln::Object< slice_image< I > >	837
mln::Image< slice_image< I > >	757

mln::Object< snake_fwd_t >	837
mln::Browsing< snake_fwd_t >	564
mln::canvas::browsing::snake_fwd_t	574
mln::Object< snake_generic_t >	837
mln::Browsing< snake_generic_t >	564
mln::canvas::browsing::snake_generic_t	575
mln::Object< snake_vert_t >	837
mln::Browsing< snake_vert_t >	564
mln::canvas::browsing::snake_vert_t	577
mln::Object< soft_heap< T, R > >	837
mln::util::soft_heap< T, R >	1071
mln::Object< sqrt >	837
mln::Function< sqrt >	710
mln::Function_v2v< sqrt >	712
mln::Object< square >	837
mln::Mesh< square >	817
mln::Regular_Grid< square >	944
mln::Object< stack_image< n, I > >	837
mln::Image< stack_image< n, I > >	757
mln::Object< static_n_face_bkd_iter< N, D > >	837
mln::Iterator< static_n_face_bkd_iter< N, D > >	781
complex_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	
complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	
mln::topo::static_n_face_bkd_iter< N, D >	1009
mln::Object< static_n_face_fwd_iter< N, D > >	837
mln::Iterator< static_n_face_fwd_iter< N, D > >	781
complex_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
mln::topo::static_n_face_fwd_iter< N, D >	1011
mln::Object< sub_image< I, S > >	837
mln::Image< sub_image< I, S > >	757
mln::Object< sub_image_if< I, S > >	837
mln::Image< sub_image_if< I, S > >	757
mln::Object< sum >	837
mln::Meta_Accumulator< sum >	818
mln::accu::meta::math::sum	471
mln::Object< sum< I, S > >	837
mln::Proxy< sum< I, S > >	941
mln::Accumulator< sum< I, S > >	538
base< S, sum< I, S > >	
mln::morpho::attribute::sum< I, S >	831
mln::Object< sum< T, S > >	837
mln::Proxy< sum< T, S > >	941
mln::Accumulator< sum< T, S > >	538
base< const S &, sum< T, S > >	
mln::accu::math::sum< T, S >	454
mln::Object< sup >	837
mln::Meta_Accumulator< sup >	818

mln::accu::meta::math::sup	472
mln::Meta_Function< sup >	820
mln::Meta_Function_vv2v< sup >	821
mln::Object< sup< T > >	837
mln::Proxy< sup< T > >	941
mln::Accumulator< sup< T > >	538
base< const T &, sup< T > >	
mln::accu::math::sup< T >	456
mln::Object< tautology >	837
mln::Function< tautology >	710
mln::Function_v2v< tautology >	712
mln::Function_v2b< tautology >	711
mln::fun::p2b::tautology	668
mln::Object< teal_t >	837
mln::Literal< teal_t >	793
mln::literal::teal_t	812
mln::Object< test >	837
mln::Function< test >	710
mln::Function_v2v< test >	712
mln::Object< threshold< V > >	837
mln::Function< threshold< V > >	710
mln::Function_v2v< threshold< V > >	712
mln::Function_v2b< threshold< V > >	711
mln::fun::v2b::threshold< V >	671
mln::Object< thru_image< I, F > >	837
mln::Image< thru_image< I, F > >	757
mln::Object< thrubin_image< I1, I2, F > >	837
mln::Image< thrubin_image< I1, I2, F > >	757
mln::Object< tick >	837
mln::Mesh< tick >	817
mln::Regular_Grid< tick >	944
mln::Object< timer >	837
mln::Proxy< timer >	941
mln::util::timer	1074
mln::Object< to16bits >	837
mln::Function< to16bits >	710
mln::Function_v2v< to16bits >	712
mln::Object< to19bits >	837
mln::Function< to19bits >	710
mln::Function_v2v< to19bits >	712
mln::Object< to23bits >	837
mln::Function< to23bits >	710
mln::Function_v2v< to23bits >	712
mln::Object< to27bits >	837
mln::Function< to27bits >	710
mln::Function_v2v< to27bits >	712
mln::Object< to8bits >	837
mln::Function< to8bits >	710

mln::Function_v2v< to8bits >	712
mln::Object< tofloat01 >	837
mln::Function< tofloat01 >	710
mln::Function_v2v< tofloat01 >	712
mln::Object< tr_image< S, I, T > >	837
mln::Image< tr_image< S, I, T > >	757
mln::Object< transformed_image< I, F > >	837
mln::Image< transformed_image< I, F > >	757
mln::Object< translation< n, C > >	837
mln::Function< translation< n, C > >	710
mln::Function_v2v< translation< n, C > >	712
mln::fun::x2x::translation< n, C >	705
mln::Object< translation_t< P > >	837
mln::Function< translation_t< P > >	710
mln::Function_v2v< translation_t< P > >	712
mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	837
mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	941
mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	538
base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	
mln::accu::tuple< A, n, >	535
mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	837
mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	818
mln::accu::meta::tuple< n, >	490
mln::Object< unary< Fun, T > >	837
mln::Function< unary< Fun, T > >	710
mln::Function_v2v< unary< Fun, T > >	712
mln::Object< unproject_image< I, D, F > >	837
mln::Image< unproject_image< I, D, F > >	757
mln::Object< up_leaf_piter< T > >	837
mln::Proxy< up_leaf_piter< T > >	941
mln::Site_Proxy< up_leaf_piter< T > >	949
mln::Site_Iterator< up_leaf_piter< T > >	947
mln::Object< up_node_piter< T > >	837
mln::Proxy< up_node_piter< T > >	941
mln::Site_Proxy< up_node_piter< T > >	949
mln::Site_Iterator< up_node_piter< T > >	947
mln::Object< up_site_piter< T > >	837
mln::Proxy< up_site_piter< T > >	941
mln::Site_Proxy< up_site_piter< T > >	949
mln::Site_Iterator< up_site_piter< T > >	947
mln::Object< val< A > >	837
mln::Proxy< val< A > >	941
mln::Accumulator< val< A > >	538
base< const A::result &, val< A > >	
mln::accu::val< A >	537
mln::Object< val< mA > >	837
mln::Meta_Accumulator< val< mA > >	818

mln::accu::meta::val< mA >	491
mln::Object< value_at_index< bool > >	837
mln::Function< value_at_index< bool > >	710
mln::Function_v2v< value_at_index< bool > >	712
mln::Object< value_at_index< T > >	837
mln::Function< value_at_index< T > >	710
mln::Function_v2v< value_at_index< T > >	712
mln::Object< var< T > >	837
mln::Proxy< var< T > >	941
mln::Accumulator< var< T > >	538
base< algebra::mat< T::dim, T::dim, float >, var< T > >	
mln::accu::stat::var< T >	531
mln::Object< variance< T, S, R > >	837
mln::Proxy< variance< T, S, R > >	941
mln::Accumulator< variance< T, S, R > >	538
base< R, variance< T, S, R > >	
mln::accu::stat::variance< T, S, R >	533
mln::Object< vec< 1, T > >	837
mln::Object< vec< 2, T > >	837
mln::Object< vec< 3, T > >	837
mln::Object< vec< 4, T > >	837
mln::Object< vec< n, C > >	837
mln::Object< vec< n, T > >	837
mln::Object< vec< V > >	837
mln::Function< vec< V > >	710
mln::Function_vv2v< vec< V > >	713
mln::fun::vv2v::vec< V >	696
mln::Object< vertex< G > >	837
mln::Site< vertex< G > >	946
mln::util::vertex< G >	1084
mln::Object< vertex_bkd_iterator< G > >	837
mln::Proxy< vertex_bkd_iterator< G > >	941
mln::Object< vertex_fwd_iterator< G > >	837
mln::Proxy< vertex_fwd_iterator< G > >	941
mln::Object< vertex_image< P, V, G > >	837
mln::Image< vertex_image< P, V, G > >	757
mln::Object< vertex_nbh_edge_bkd_iterator< G > >	837
mln::Proxy< vertex_nbh_edge_bkd_iterator< G > >	941
mln::Object< vertex_nbh_edge_fwd_iterator< G > >	837
mln::Proxy< vertex_nbh_edge_fwd_iterator< G > >	941
mln::Object< vertex_nbh_vertex_bkd_iterator< G > >	837
mln::Proxy< vertex_nbh_vertex_bkd_iterator< G > >	941
mln::Object< vertex_nbh_vertex_fwd_iterator< G > >	837
mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > >	941
mln::Object< violent_cast_image< T, I > >	837
mln::Image< violent_cast_image< T, I > >	757
mln::Object< violet_t >	837
mln::Literal< violet_t >	793

mln::literal::violet_t	813
mln::Object< viota_t >	837
mln::Function< viota_t >	710
mln::Function_v2v< viota_t >	712
mln::Object< viota_t< S > >	837
mln::Function< viota_t< S > >	710
mln::Function_v2v< viota_t< S > >	712
mln::Object< volume >	837
mln::Meta_Accumulator< volume >	818
mln::accu::meta::shape::volume	480
mln::Object< volume< I > >	837
mln::Proxy< volume< I > >	941
mln::Accumulator< volume< I > >	538
base< unsigned, volume< I > >	
mln::accu::shape::volume< I >	503
mln::morpho::attribute::volume< I >	833
mln::Object< W >	837
mln::Object< w_window< D, W > >	837
mln::Weighted_Window< w_window< D, W > >	1135
weighted_window_base< mln::window< D >, w_window< D, W > >	
mln::w_window< D, W >	1132
mln::Object< white_gaussian< V > >	837
mln::Function< white_gaussian< V > >	710
mln::Function_n2v< white_gaussian< V > >	711
mln::fun::n2v::white_gaussian< V >	666
mln::Object< white_t >	837
mln::Literal< white_t >	793
mln::literal::white_t	814
mln::Object< window< D > >	837
mln::Window< window< D > >	1148
window_base< D, window< D > >	
mln::window< D >	1149
mln::Object< wrap >	837
mln::Function< wrap >	710
mln::Function_v2v< wrap >	712
mln::Object< wrap< L > >	837
mln::Function< wrap< L > >	710
mln::Function_v2v< wrap< L > >	712
mln::Object< yellow_t >	837
mln::Literal< yellow_t >	793
mln::literal::yellow_t	815
mln::Object< yes >	837
mln::util::yes	1089
mln::Object< zero_t >	837
mln::Literal< zero_t >	793
mln::literal::zero_t	816
mln::internal::pixel_impl_< I, bkd_pixter1d< I > >	
pixel_iterator_base_< I, bkd_pixter1d< I > >	

mln::internal::pixel_impl_< I, bkd_pixter2d< I > >	
pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::internal::pixel_impl_< I, bkd_pixter3d< I > >	
pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::internal::pixel_impl_< I, dpoints_bkd_pixter< I > >	
mln::dpoints_bkd_pixter< I >	644
mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I >	646
mln::internal::pixel_impl_< I, fwd_pixter1d< I > >	
pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter2d< I > >	
pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter3d< I > >	
pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::internal::pixel_impl_< I, pixel< I > >	
mln::pixel< I >	928
trait::graph< I >	1155
trait::graph< mln::complex_image< I, G, V > >	1156
trait::graph< mln::image2d< T > >	1156

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

mln::accu::center< P, V > (Mass center accumulator)	433
mln::accu::convolve< T1, T2, R > (Generic convolution accumulator class)	435
mln::accu::count_adjacent_vertices< F, S > (Accumulator class counting the number of vertices adjacent to a set of mln::p_edges_psite (i.e., a set of edges))	436
mln::accu::count_labels< L > (Count the number of different labels in an image)	438
mln::accu::count_value< V > (Define an accumulator that counts the occurrence of a given value)	439
mln::accu::histo< V > (Generic histogram class over a value set with type V)	441
mln::accu::label_used< L > (References all the labels used)	442
mln::accu::logic::land ("Logical-and" accumulator)	444
mln::accu::logic::land_basic ("Logical-and" accumulator)	445
mln::accu::logic::lor ("Logical-or" accumulator)	447
mln::accu::logic::lor_basic ("Logical-or" accumulator class)	448
mln::accu::maj_h< T > (Compute the majority value)	450
mln::accu::math::count< T > (Generic counter accumulator)	451
mln::accu::math::inf< T > (Generic inf accumulator class)	453
mln::accu::math::sum< T, S > (Generic sum accumulator class)	454
mln::accu::math::sup< T > (Generic sup accumulator class)	456
mln::accu::max_site< I > (Define an accumulator that computes the first site with the maximum value in an image)	457
mln::accu::meta::center (Meta accumulator for center)	459
mln::accu::meta::count_adjacent_vertices (Meta accumulator for count_adjacent_vertices) . . .	459
mln::accu::meta::count_labels (Meta accumulator for count_labels)	460
mln::accu::meta::count_value (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value)	461
mln::accu::meta::histo (Meta accumulator for histo)	462
mln::accu::meta::label_used (Meta accumulator for label_used)	463
mln::accu::meta::logic::land (Meta accumulator for land)	464
mln::accu::meta::logic::land_basic (Meta accumulator for land_basic)	465
mln::accu::meta::logic::lor (Meta accumulator for lor)	466
mln::accu::meta::logic::lor_basic (Meta accumulator for lor_basic)	467
mln::accu::meta::maj_h (Meta accumulator for maj_h)	468
mln::accu::meta::math::count (Meta accumulator for count)	469
mln::accu::meta::math::inf (Meta accumulator for inf)	470

mln::accu::meta::math::sum (Meta accumulator for sum)	471
mln::accu::meta::math::sup (Meta accumulator for sup)	472
mln::accu::meta::max_site (Meta accumulator for max_site)	473
mln::accu::meta::nil (Meta accumulator for nil)	474
mln::accu::meta::p< mA > (Meta accumulator for p)	475
mln::accu::meta::pair< A1, A2 > (Meta accumulator for pair)	476
mln::accu::meta::rms (Meta accumulator for rms)	477
mln::accu::meta::shape::bbox (Meta accumulator for bbox)	478
mln::accu::meta::shape::height (Meta accumulator for height)	479
mln::accu::meta::shape::volume (Meta accumulator for volume)	480
mln::accu::meta::stat::max (Meta accumulator for max)	481
mln::accu::meta::stat::max_h (Meta accumulator for max)	482
mln::accu::meta::stat::mean (Meta accumulator for mean)	483
mln::accu::meta::stat::median_alt< T > (Meta accumulator for median_alt)	484
mln::accu::meta::stat::median_h (Meta accumulator for median_h)	485
mln::accu::meta::stat::min (Meta accumulator for min)	486
mln::accu::meta::stat::min_h (Meta accumulator for min)	487
mln::accu::meta::stat::rank (Meta accumulator for rank)	488
mln::accu::meta::stat::rank_high_quant (Meta accumulator for rank_high_quant)	489
mln::accu::meta::tuple< n, > (Meta accumulator for tuple)	490
mln::accu::meta::val< mA > (Meta accumulator for val)	491
mln::accu::nil< T > (Define an accumulator that does nothing)	492
mln::accu::p< A > (Generic p of accumulators)	494
mln::accu::pair< A1, A2, T > (Generic pair of accumulators)	495
mln::accu::rms< T, V > (Generic root mean square accumulator class)	498
mln::accu::shape::bbox< P > (Generic bounding box accumulator class)	500
mln::accu::shape::height< I > (Height accumulator)	501
mln::accu::shape::volume< I > (Volume accumulator class)	503
mln::accu::site_set::rectangularity< P > (Compute the rectangularity of a site set)	505
mln::accu::stat::deviation< T, S, M > (Generic standard deviation accumulator class)	507
mln::accu::stat::histo3d_rgb< V > (Define a histogram as accumulator which returns an image3d)	509
mln::accu::stat::max< T > (Generic max accumulator class)	511
mln::accu::stat::max_h< V > (Generic max function based on histogram over a value set with type V)	513
mln::accu::stat::mean< T, S, M > (Generic mean accumulator class)	514
mln::accu::stat::median_alt< S > (Generic median_alt function based on histogram over a value set with type S)	516
mln::accu::stat::median_h< V > (Generic median function based on histogram over a value set with type V)	518
mln::accu::stat::meta::deviation (Meta accumulator for deviation)	519
mln::accu::stat::min< T > (Generic min accumulator class)	520
mln::accu::stat::min_h< V > (Generic min function based on histogram over a value set with type V)	522
mln::accu::stat::min_max< V > (Generic min and max accumulator class)	523
mln::accu::stat::rank< T > (Generic rank accumulator class)	526
mln::accu::stat::rank< bool > (Rank accumulator class for Boolean)	528
mln::accu::stat::rank_high_quant< T > (Generic rank accumulator class)	529
mln::accu::stat::var< T > (Var accumulator class)	531
mln::accu::stat::variance< T, S, R > (Variance accumulator class)	533
mln::accu::tuple< A, n, > (Generic tuple of accumulators)	535
mln::accu::val< A > (Generic val of accumulators)	537
mln::Accumulator< E > (Base class for implementation of accumulators)	538
mln::algebra::h_mat< d, T > (N-Dimensional matrix with homogeneous coordinates)	540
mln::algebra::h_vec< d, C > (N-Dimensional vector with homogeneous coordinates)	542

mln::bkd_pixter1d< I > (Backward pixel iterator on a 1-D image with border)	544
mln::bkd_pixter2d< I > (Backward pixel iterator on a 2-D image with border)	545
mln::bkd_pixter3d< I > (Backward pixel iterator on a 3-D image with border)	547
mln::box< P > (Generic box class: site set containing points of a regular grid)	548
mln::Box< E > (Base class for implementation classes of boxes)	556
mln::box_runend_piter< P > (A generic backward iterator on points by lines)	562
mln::box_runstart_piter< P > (A generic forward iterator on points by lines)	563
mln::Browsing< E > (Base class for implementation classes that are browsings)	564
mln::canvas::browsing::backdiagonal2d_t (Browsing in a certain direction)	565
mln::canvas::browsing::breadth_first_search_t (Breadth-first search algorithm for graph, on vertices)	566
mln::canvas::browsing::depth_first_search_t (Breadth-first search algorithm for graph, on vertices)	566
mln::canvas::browsing::diagonal2d_t (Browsing in a certain direction)	567
mln::canvas::browsing::dir_struct_elt_incr_update_t (Browsing in a certain direction with a segment)	568
mln::canvas::browsing::directional_t (Browsing in a certain direction)	569
mln::canvas::browsing::fwd_t (Canvas for forward browsing)	571
mln::canvas::browsing::hyper_directional_t (Browsing in a certain direction)	572
mln::canvas::browsing::snake_fwd_t (Browsing in a snake-way, forward)	574
mln::canvas::browsing::snake_generic_t (Multidimensional Browsing in a given-way)	575
mln::canvas::browsing::snake_vert_t (Browsing in a snake-way, forward)	577
mln::canvas::chamfer< F > (Compute chamfer distance)	578
mln::category< R(*) (A) > (Category declaration for a unary C function)	578
mln::complex_image< D, G, V > (Image based on a complex)	579
mln::complex_neighborhood_bkd_piter< I, G, N > (Backward iterator on complex neighborhood)	582
mln::complex_neighborhood_fwd_piter< I, G, N > (Forward iterator on complex neighborhood)	583
mln::complex_psite< D, G > (Point site associated to a mln::p_complex)	585
mln::complex_window_bkd_piter< I, G, W > (Backward iterator on complex window)	588
mln::complex_window_fwd_piter< I, G, W > (Forward iterator on complex window)	590
mln::decorated_image< I, D > (Image that can have additional features)	591
mln::Delta_Point_Site< E > (FIXME: Doc!)	594
mln::Delta_Point_Site< void > (Delta point site category flag type)	595
mln::doc::Accumulator< E > (Documentation class for mln::Accumulator)	595
mln::doc::Box< E > (Documentation class for mln::Box)	596
mln::doc::Dpoint< E > (Documentation class for mln::Dpoint)	599
mln::doc::Fastest_Image< E > (Documentation class for the concept of images that have the speed property set to "fastest")	601
mln::doc::Generalized_Pixel< E > (Documentation class for mln::Generalized_Pixel)	609
mln::doc::Image< E > (Documentation class for mln::Image)	611
mln::doc::Iterator< E > (Documentation class for mln::Iterator)	617
mln::doc::Neighborhood< E > (Documentation class for mln::Neighborhood)	618
mln::doc::Object< E > (Documentation class for mln::Object)	620
mln::doc::Pixel_Iterator< E > (Documentation class for mln::Iterator)	620
mln::doc::Point_Site< E > (Documentation class for mln::Point_Site)	623
mln::doc::Site_Iterator< E > (Documentation class for mln::Site_Iterator)	626
mln::doc::Site_Set< E > (Documentation class for mln::Site_Set)	628
mln::doc::Value_Iterator< E > (Documentation class for mln::Value_Iterator)	630
mln::doc::Value_Set< E > (Documentation class for mln::Value_Set)	631
mln::doc::Weighted_Window< E > (Documentation class for mln::Weighted_Window)	634
mln::doc::Window< E > (Documentation class for mln::Window)	636
mln::Dpoint< E > (Base class for implementation of delta-point classes)	638
mln::dpoint< G, C > (Generic delta-point class)	639

mln::dpoints_bkd_pixter< I > (A generic backward iterator on the pixels of a dpoint-based window or neighborhood)	644
mln::dpoints_fwd_pixter< I > (A generic forward iterator on the pixels of a dpoint-based window or neighborhood)	646
mln::dpsites_bkd_piter< V > (A generic backward iterator on points of windows and of neighborhoods)	649
mln::dpsites_fwd_piter< V > (A generic forward iterator on points of windows and of neighborhoods)	650
mln::Edge< E > (Edge category flag type)	652
mln::edge_image< P, V, G > (Image based on graph edges)	652
mln::extended< I > (Makes an image become restricted by a point set)	654
mln::extension_fun< I, F > (Extends the domain of an image with a function)	656
mln::extension_ima< I, J > (Extends the domain of an image with an image)	658
mln::extension_val< I > (Extends the domain of an image with a value)	661
mln::flat_image< T, S > (Image with a single value)	663
mln::fun::from_accu< A > (Wrap an accumulator into a function)	666
mln::fun::n2v::white_gaussian< V > (Generate a White Gaussian Noise)	666
mln::fun::p2b::antilogy (A p2b function always returning false)	667
mln::fun::p2b::tautology (A p2b function always returning true)	668
mln::fun::v2b::lnot< V > (Functor computing logical-not on a value)	670
mln::fun::v2b::threshold< V > (Threshold function)	671
mln::fun::v2v::ch_function_value< F, V > (Wrap a function v2v and convert its result to another type)	672
mln::fun::v2v::component< T, i > (Functor that accesses the i-th component of a value)	672
mln::fun::v2v::l1_norm< V, R > (L1-norm)	673
mln::fun::v2v::l2_norm< V, R > (L2-norm)	674
mln::fun::v2v::linear< V, T, R > (Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type)	675
mln::fun::v2v::linfty_norm< V, R > (L-infty norm)	676
mln::fun::v2v::rgb8_to_rgn< n > (Convert a rgb8 value to a rgn, $n < 8$)	677
mln::fun::v2w2v::cos< V > (Cosinus bijective functor)	679
mln::fun::v2w_w2v::l1_norm< V, R > (L1-norm)	680
mln::fun::v2w_w2v::l2_norm< V, R > (L2-norm)	682
mln::fun::v2w_w2v::linfty_norm< V, R > (L-infty norm)	683
mln::fun::vv2b::eq< L, R > (Functor computing equal between two values)	684
mln::fun::vv2b::ge< L, R > (Functor computing "greater or equal than" between two values)	684
mln::fun::vv2b::gt< L, R > (Functor computing "greater than" between two values)	685
mln::fun::vv2b::implies< L, R > (Functor computing logical-implies between two values)	686
mln::fun::vv2b::le< L, R > (Functor computing "lower or equal than" between two values)	687
mln::fun::vv2b::lt< L, R > (Functor computing "lower than" between two values)	688
mln::fun::vv2v::diff_abs< V > (A functor computing the diff_absimum of two values)	689
mln::fun::vv2v::land< L, R > (Functor computing logical-and between two values)	690
mln::fun::vv2v::land_not< L, R > (Functor computing logical and-not between two values)	691
mln::fun::vv2v::lor< L, R > (Functor computing logical-or between two values)	692
mln::fun::vv2v::lxor< L, R > (Functor computing logical-xor between two values)	693
mln::fun::vv2v::max< V > (A functor computing the maximum of two values)	694
mln::fun::vv2v::min< L, R > (A functor computing the minimum of two values)	695
mln::fun::vv2v::vec< V > (A functor computing the vecimum of two values)	696
mln::fun::x2p::closest_point< P > (FIXME: doxygen + concept checking)	697
mln::fun::x2v::bilinear< I > (Represent a bilinear interolation of values from an underlying image)	698
mln::fun::x2v::trilinear< I > (Represent a trilinear interolation of values from an underlying image)	699
mln::fun::x2x::composed< T2, T1 > (Represent a composition of two transformations)	699

<code>mln::fun::x2x::linear< I ></code> (Represent a linear interolation of values from an underlying image)	700
<code>mln::fun::x2x::rotation< n, C ></code> (Represent a rotation function)	701
<code>mln::fun::x2x::translation< n, C ></code> (Translation function-object)	705
<code>mln::fun_image< F, I ></code> (Image read through a function)	707
<code>mln::Function< E ></code> (Base class for implementation of function-objects)	710
<code>mln::Function< void ></code> (Function category flag type)	710
<code>mln::Function_n2v< E ></code> (Base class for implementation of function-objects from Nil to value)	711
<code>mln::Function_v2b< E ></code> (Base class for implementation of function-objects from a value to a Boolean)	711
<code>mln::Function_v2v< E ></code> (Base class for implementation of function-objects from value to value)	712
<code>mln::Function_vv2b< E ></code> (Base class for implementation of function-objects from a couple of values to a Boolean)	713
<code>mln::Function_vv2v< E ></code> (Base class for implementation of function-objects from a couple of values to a value)	713
<code>mln::fwd_pixter1d< I ></code> (Forward pixel iterator on a 1-D image with border)	714
<code>mln::fwd_pixter2d< I ></code> (Forward pixel iterator on a 2-D image with border)	715
<code>mln::fwd_pixter3d< I ></code> (Forward pixel iterator on a 3-D image with border)	717
<code>mln::Gdpoint< E ></code> (FIXME: Doc!)	718
<code>mln::Gdpoint< void ></code> (Delta point site category flag type)	719
<code>mln::Generalized_Pixel< E ></code> (Base class for implementation classes that are pixels or that have the behavior of pixels)	720
<code>mln::geom::complex_geometry< D, P ></code> (A functor returning the sites of the faces of a complex where the locations of each 0-face is stored)	720
<code>mln::Gpoint< E ></code> (Base class for implementation of point classes)	722
<code>mln::Graph< E ></code> (Base class for implementation of graph classes)	726
<code>mln::graph::attribute::card_t</code> (Compute the cardinality of every component in a graph)	727
<code>mln::graph::attribute::representative_t</code> (Compute the representative vertex of every component in a graph)	727
<code>mln::graph_elt_mixed_neighborhood< G, S, S2 ></code> (Elementary neighborhood on graph class)	728
<code>mln::graph_elt_mixed_window< G, S, S2 ></code> (Elementary window on graph class)	729
<code>mln::graph_elt_neighborhood< G, S ></code> (Elementary neighborhood on graph class)	733
<code>mln::graph_elt_neighborhood_if< G, S, I ></code> (Elementary neighborhood_if on graph class)	735
<code>mln::graph_elt_window< G, S ></code> (Elementary window on graph class)	737
<code>mln::graph_elt_window_if< G, S, I ></code> (Custom window on graph class)	741
<code>mln::graph_window_base< P, E ></code>	746
<code>mln::graph_window_if_piter< S, W, I ></code> (Forward iterator on line graph window)	748
<code>mln::graph_window_piter< S, W, I ></code> (Forward iterator on line graph window)	750
<code>mln::hexa< I ></code> (Hexagonal image class)	753
<code>mln::histo::array< T ></code> (Generic histogram class over a value set with type T)	757
<code>mln::Image< E ></code> (Base class for implementation of image classes)	757
<code>mln::image1d< T ></code> (Basic 1D image class)	759
<code>mln::image2d< T ></code> (Basic 2D image class)	764
<code>mln::image2d_h< V ></code> (2d image based on an hexagonal mesh)	769
<code>mln::image3d< T ></code> (Basic 3D image class)	772
<code>mln::interpolated< I, F ></code> (Makes the underlying image being accessed with floating coordinates)	777
<code>mln::io::dicom::dicom_header</code> (Store dicom file header)	780
<code>mln::io::dump::dump_header</code> (Store dump file header)	780
<code>mln::io::fld::fld_header</code> (Define the header structure of an AVS field data file)	780
<code>mln::io::raw::raw_header</code> (Store raw file header)	780
<code>mln::Iterator< E ></code> (Base class for implementation classes that are iterators)	781
<code>mln::labeled_image< I ></code> (Morpher providing an improved interface for labeled image)	783
<code>mln::labeled_image_base< I, E ></code> (Base class Morpher providing an improved interface for labeled image)	787
<code>mln::lazy_image< I, F, B ></code> (Image values are computed on the fly)	790

mln::Literal< E > (Base class for implementation classes of literals)	793
mln::literal::black_t (Type of literal black)	795
mln::literal::blue_t (Type of literal blue)	796
mln::literal::brown_t (Type of literal brown)	796
mln::literal::cyan_t (Type of literal cyan)	797
mln::literal::green_t (Type of literal green)	798
mln::literal::identity_t (Type of literal identity)	799
mln::literal::light_gray_t (Type of literal grays)	800
mln::literal::lime_t (Type of literal lime)	801
mln::literal::magenta_t (Type of literal magenta)	802
mln::literal::max_t (Type of literal max)	803
mln::literal::min_t (Type of literal min)	804
mln::literal::olive_t (Type of literal olive)	805
mln::literal::one_t (Type of literal one)	806
mln::literal::orange_t (Type of literal orange)	807
mln::literal::origin_t (Type of literal origin)	808
mln::literal::pink_t (Type of literal pink)	809
mln::literal::purple_t (Type of literal purple)	810
mln::literal::red_t (Type of literal red)	811
mln::literal::teal_t (Type of literal teal)	812
mln::literal::violet_t (Type of literal violet)	813
mln::literal::white_t (Type of literal white)	814
mln::literal::yellow_t (Type of literal yellow)	815
mln::literal::zero_t (Type of literal zero)	816
mln::Mesh< E > (Base class for implementation classes of meshes)	817
mln::Meta_Accumulator< E > (Base class for implementation of meta accumulators)	818
mln::Meta_Function< E > (Base class for implementation of meta functions)	820
mln::Meta_Function_v2v< E > (Base class for implementation of function-objects from value to value)	821
mln::Meta_Function_vv2v< E > (Base class for implementation of function-objects from value to value)	821
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > (Ands type)	822
mln::metal::converts_to< T, U > ("converts-to" check)	822
mln::metal::equal< T1, T2 > (Definition of a static 'equal' test)	822
mln::metal::goes_to< T, U > ("goes-to" check)	823
mln::metal::is< T, U > ("is" check)	823
mln::metal::is_a< T, M > ("is_a" check)	824
mln::metal::is_not< T, U > ("is_not" check)	824
mln::metal::is_not_a< T, M > ("is_not_a" static Boolean expression)	824
mln::morpho::attribute::card< I > (Cardinality accumulator class)	824
mln::morpho::attribute::count_adjacent_vertices< I > (Count_Adjacent_Vertices accumulator class)	826
mln::morpho::attribute::height< I > (Height accumulator class)	827
mln::morpho::attribute::sharpness< I > (Sharpness accumulator class)	829
mln::morpho::attribute::sum< I, S > (Suminality accumulator class)	831
mln::morpho::attribute::volume< I > (Volume accumulator class)	833
mln::neighb< W > (Adapter class from window to neighborhood)	835
mln::Neighborhood< E > (Base class for implementation classes that are neighborhoods)	836
mln::Neighborhood< void > (Neighborhood category flag type)	837
mln::Object< E > (Base class for almost every class defined in Milena)	837
mln::p2p_image< I, F > (FIXME: Doc!)	837
mln::p_array< P > (Multi-set of sites)	839
mln::p_centered< W > (Site set corresponding to a window centered on a site)	845

<code>mln::p_complex< D, G ></code> (A complex psite set based on the N-faces of a complex of dimension D (a D-complex))	848
<code>mln::p_edges< G, F ></code> (Site set mapping graph edges and image sites)	852
<code>mln::p_faces< N, D, P ></code> (A complex psite set based on a the N-faces of a complex of dimension D (a D-complex))	858
<code>mln::p_graph_piter< S, I ></code> (Generic iterator on point sites of a <code>mln::S</code>)	861
<code>mln::p_if< S, F ></code> (Site set restricted w.r.t)	862
<code>mln::p_image< I ></code> (Site set based on an image of Booleans)	866
<code>mln::p_indexed_bkd_piter< S ></code> (Backward iterator on sites of an indexed site set)	870
<code>mln::p_indexed_fwd_piter< S ></code> (Forward iterator on sites of an indexed site set)	871
<code>mln::p_indexed_psite< S ></code> (Psite class for indexed site sets such as <code>p_array</code>)	873
<code>mln::p_key< K, P ></code> (Priority queue class)	873
<code>mln::p_line2d</code> (2D discrete line of points)	878
<code>mln::p_mutable_array_of< S ></code> (<code>P_mutable_array_of</code> is a mutable array of site sets)	882
<code>mln::p_n_faces_bkd_piter< D, G ></code> (Backward iterator on the n-faces sites of an <code>mln::p_complex<D, G></code>)	886
<code>mln::p_n_faces_fwd_piter< D, G ></code> (Forward iterator on the n-faces sites of an <code>mln::p_complex<D, G></code>)	887
<code>mln::p_priority< P, Q ></code> (Priority queue)	889
<code>mln::p_queue< P ></code> (Queue of sites (based on <code>std::deque</code>))	894
<code>mln::p_queue_fast< P ></code> (Queue of sites class (based on <code>p_array</code>))	899
<code>mln::p_run< P ></code> (Point set class in run)	904
<code>mln::p_set< P ></code> (Mathematical set of sites (based on <code>util::set</code>))	909
<code>mln::p_transformed< S, F ></code> (Site set transformed through a function)	913
<code>mln::p_transformed_piter< Pi, S, F ></code> (Iterator on <code>p_transformed<S,F></code>)	916
<code>mln::p_vaccess< V, S ></code> (Site set in which sites are grouped by their associated value)	918
<code>mln::p_vertices< G, F ></code> (Site set based mapping graph vertices to sites)	922
<code>mln::pixel< I ></code> (Generic pixel class)	928
<code>mln::plain< I ></code> (Prevents an image from sharing its data)	930
<code>mln::Point< P ></code> (Base class for implementation of point classes)	932
<code>mln::point< G, C ></code> (Generic point class)	934
<code>mln::Proxy< E ></code> (Base class for implementation classes of the notion of "proxy")	941
<code>mln::Proxy< void ></code> (Proxy category flag type)	941
<code>mln::Pseudo_Site< E ></code> (Base class for implementation classes of the notion of "pseudo site")	941
<code>mln::Pseudo_Site< void ></code> (Pseudo_Site category flag type)	942
<code>mln::pw::image< F, S ></code> (A generic point-wise image implementation)	942
<code>mln::registration::closest_point_basic< P ></code> (Closest point functor based on map distance)	943
<code>mln::registration::closest_point_with_map< P ></code> (Closest point functor based on map distance)	944
<code>mln::Regular_Grid< E ></code> (Base class for implementation classes of regular grids)	944
<code>mln::safe_image< I ></code> (Makes an image accessible at undefined location)	945
<code>mln::select::p_of< P ></code> (Structure <code>p_of</code>)	945
<code>mln::Site< E ></code> (Base class for classes that are explicitly sites)	946
<code>mln::Site< void ></code> (Site category flag type)	947
<code>mln::Site_Iterator< E ></code> (Base class for implementation of classes of iterator on points)	947
<code>mln::Site_Proxy< E ></code> (Base class for implementation classes of the notion of "site proxy")	949
<code>mln::Site_Proxy< void ></code> (Site_Proxy category flag type)	950
<code>mln::Site_Set< E ></code> (Base class for implementation classes of site sets)	950
<code>mln::Site_Set< void ></code> (Site_Set category flag type)	953
<code>mln::sub_image< I, S ></code> (Image having its domain restricted by a site set)	953
<code>mln::sub_image_if< I, S ></code> (Image having its domain restricted by a site set and a function)	955
<code>mln::thru_image< I, F ></code> (Morph image values through a function)	956
<code>mln::thrubin_image< I1, I2, F ></code> (Morphes values from two images through a binary function)	957

mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > (Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>)	959
mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > (Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>)	960
mln::topo::adj_higher_face_bkd_iter< D > (Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>)	961
mln::topo::adj_higher_face_fwd_iter< D > (Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>)	962
mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > (Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>)	963
mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > (Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>)	964
mln::topo::adj_lower_face_bkd_iter< D > (Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>)	965
mln::topo::adj_lower_face_fwd_iter< D > (Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>)	967
mln::topo::adj_lower_higher_face_bkd_iter< D > (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>)	968
mln::topo::adj_lower_higher_face_fwd_iter< D > (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>)	969
mln::topo::adj_m_face_bkd_iter< D > (Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex)	970
mln::topo::adj_m_face_fwd_iter< D > (Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex)	971
mln::topo::algebraic_face< D > (Algebraic face handle in a complex; the face dimension is dynamic)	973
mln::topo::algebraic_n_face< N, D > (Algebraic N-face handle in a complex)	978
mln::topo::center_only_iter< D > (Iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>)	982
mln::topo::centered_bkd_iter_adapter< D, I > (Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces)	984
mln::topo::centered_fwd_iter_adapter< D, I > (Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces)	985
mln::topo::complex< D > (General complex of dimension D)	986
mln::topo::face< D > (Face handle in a complex; the face dimension is dynamic)	989
mln::topo::face_bkd_iter< D > (Backward iterator on all the faces of an mln::complex<D>)	993
mln::topo::face_fwd_iter< D > (Forward iterator on all the faces of an mln::complex<D>)	995
mln::topo::is_n_face< N > (A functor testing wheter a mln::complex_psite is an N-face)	996
mln::topo::is_simple_2d_t< N > (Test if a point is simple or not)	997
mln::topo::is_simple_cell< I > (A predicate for the simplicity of a point based on the collapse property of the attachment)	998
mln::topo::n_face< N, D > (N-face handle in a complex)	1001
mln::topo::n_face_bkd_iter< D > (Backward iterator on all the faces of an mln::complex<D>)	1005
mln::topo::n_face_fwd_iter< D > (Forward iterator on all the faces of an mln::complex<D>)	1006
mln::topo::n_faces_set< N, D > (Set of face handles of dimension N)	1007
mln::topo::skeleton::is_simple_point< N >	1009
mln::topo::static_n_face_bkd_iter< N, D > (Backward iterator on all the N-faces of a mln::complex<D>)	1009
mln::topo::static_n_face_fwd_iter< N, D > (Forward iterator on all the N-faces of a mln::complex<D>)	1011

<code>mln::tr_image< S, I, T ></code> (Transform an image by a given transformation)	1012
<code>mln::transformed_image< I, F ></code> (Image having its domain restricted by a site set)	1015
<code>mln::unproject_image< I, D, F ></code> (Un-projects an image)	1017
<code>mln::util::adjacency_matrix< V ></code> (A class of adjacency matrix)	1019
<code>mln::util::array< T ></code> (A dynamic array class)	1019
<code>mln::util::branch< T ></code> (Class of generic branch)	1027
<code>mln::util::branch_iter< T ></code> (Basic 2D image class)	1028
<code>mln::util::branch_iter_ind< T ></code> (Basic 2D image class)	1030
<code>mln::util::couple< T, U ></code> (Definition of a couple)	1031
<code>mln::util::eat</code> (Eat structure)	1033
<code>mln::util::edge< G ></code> (Edge of a graph <i>G</i>)	1034
<code>mln::util::fibonacci_heap< P, T ></code> (Fibonacci heap)	1038
<code>mln::util::graph</code> (Undirected graph)	1042
<code>mln::util::greater_point< I ></code> (A “greater than” functor comparing points w.r.t)	1048
<code>mln::util::greater_psite< I ></code> (A “greater than” functor comparing psites w.r.t)	1049
<code>mln::util::head< T, R ></code> (Top structure of the soft heap)	1049
<code>mln::util::ignore</code> (Ignore structure)	1050
<code>mln::util::ilcell< T ></code> (Element of an item list. Store the data (key) used in soft_heap)	1050
<code>mln::util::line_graph< G ></code> (Undirected line graph of a graph of type <i>G</i>)	1051
<code>mln::util::nil</code> (Nil structure)	1056
<code>mln::util::node< T, R ></code> (Meta-data of an element in the heap)	1057
<code>mln::util::object_id< Tag, V ></code> (Base class of an object id)	1057
<code>mln::util::ord< T ></code> (Function-object that defines an ordering between objects with type <i>T</i> : <i>lhs</i> <i>R rhs</i>)	1059
<code>mln::util::ord_pair< T ></code> (Ordered pair structure s.a)	1059
<code>mln::util::pix< I ></code> (Structure <i>pix</i>)	1062
<code>mln::util::set< T ></code> (An “efficient” mathematical set class)	1063
<code>mln::util::site_pair< P ></code> (A pair of sites)	1069
<code>mln::util::soft_heap< T, R ></code> (Soft heap)	1071
<code>mln::util::timer</code> (Timer structure)	1074
<code>mln::util::tracked_ptr< T ></code> (Smart pointer for shared data with tracking)	1075
<code>mln::util::tree< T ></code> (Class of generic tree)	1077
<code>mln::util::tree_node< T ></code> (Class of generic tree_node for tree)	1079
<code>mln::util::vertex< G ></code> (Vertex of a graph <i>G</i>)	1084
<code>mln::util::yes</code> (Object that always says “yes”)	1089
<code>mln::Value< E ></code> (Base class for implementation classes of values)	1089
<code>mln::value::float01</code> (Class for floating values restricted to the interval [0..1] and discretized with <i>n</i> bits)	1091
<code>mln::value::float01_f</code> (Class for floating values restricted to the interval [0..1])	1093
<code>mln::value::graylevel< n ></code> (General gray-level class on <i>n</i> bits)	1094
<code>mln::value::graylevel_f</code> (General gray-level class on <i>n</i> bits)	1098
<code>mln::value::int_s< n ></code> (Signed integer value class)	1100
<code>mln::value::int_u< n ></code> (Unsigned integer value class)	1103
<code>mln::value::int_u_sat< n ></code> (Unsigned integer value class with saturation behavior)	1105
<code>mln::value::Integer< E ></code> (Concept of integer)	1107
<code>mln::value::Integer< void ></code> (Category flag type)	1108
<code>mln::value::label< n ></code> (Label value class)	1108
<code>mln::value::lut_vec< S, T ></code> (Class that defines FIXME)	1111
<code>mln::value::proxy< I ></code> (Generic proxy class for an image pixel value)	1114
<code>mln::value::qt::rgb32</code> (Color class for red-green-blue where every component is <i>n</i> -bit encoded)	1116
<code>mln::value::rgb< n ></code> (Color class for red-green-blue where every component is <i>n</i> -bit encoded)	1118
<code>mln::value::set< T ></code> (Class that defines the set of values of type <i>T</i>)	1120
<code>mln::value::sign</code> (Value type composed by the set {−1, 0, 1} sign value type is a subset of the int value type)	1121

<code>mln::value::stack_image< n, I ></code> (Stack image class)	1123
<code>mln::value::super_value< sign ></code> (Specializations:)	1126
<code>mln::value::value_array< T, V ></code> (Generic array class over indexed by a value set with type <code>T</code>)	1126
<code>mln::Vertex< E ></code> (<code>Vertex</code> category flag type)	1127
<code>mln::vertex_image< P, V, G ></code> (<code>Image</code> based on graph vertices)	1128
<code>mln::violent_cast_image< T, I ></code> (Violently cast image values to a given type)	1130
<code>mln::w_window< D, W ></code> (Generic <code>w_window</code> class)	1132
<code>mln::Weighted_Window< E ></code> (Base class for implementation classes that are weighted- windows)	1135
<code>mln::win::backdiag2d</code> (Diagonal line window defined on the 2D square grid)	1137
<code>mln::win::ball< G, C ></code> (Generic ball window defined on a given grid)	1138
<code>mln::win::cube3d</code> (Cube window defined on the 3D grid)	1139
<code>mln::win::cuboid3d</code> (Cuboid defined on the 3-D square grid)	1140
<code>mln::win::diag2d</code> (Diagonal line window defined on the 2D square grid)	1142
<code>mln::win::line< M, i, C ></code> (Generic line window defined on a given grid in the given dimension)	1143
<code>mln::win::multiple< W, F ></code> (Multiple window)	1145
<code>mln::win::multiple_size< n, W, F ></code> (Definition of a multiple-size window)	1145
<code>mln::win::octagon2d</code> (Octagon window defined on the 2D square grid)	1145
<code>mln::win::rectangle2d</code> (Rectangular window defined on the 2D square grid)	1147
<code>mln::Window< E ></code> (Base class for implementation classes that are windows)	1148
<code>mln::window< D ></code> (Generic window class)	1149
<code>mln::world::inter_pixel::is_separator</code> (Functor returning whether a site is a separator in an inter- pixel image)	1154
<code>trait::graph< I ></code> (Graph traits)	1155
<code>trait::graph< mln::complex_image< l, G, V > ></code> (Graph traits for l-complexes images)	1156
<code>trait::graph< mln::image2d< T > ></code> (Graph traits for <code>mln::image2d</code>)	1156

Chapter 8

Module Documentation

8.1 On site sets

Accumulators working on site sets.

Classes

- struct `mln::accu::center< P, V >`
Mass center accumulator.
- struct `mln::accu::math::count< T >`
Generic counter accumulator.
- struct `mln::accu::shape::bbox< P >`
Generic bounding box accumulator class.
- class `mln::accu::site_set::rectangularity< P >`
Compute the rectangularity of a site set.

8.1.1 Detailed Description

Accumulators working on site sets.

8.2 On images

Accumulators working on images.

Classes

- struct `mln::accu::count_adjacent_vertices< F, S >`
Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

- struct `mln::accu::max_site< I >`
Define an accumulator that computes the first site with the maximum value in an image.
- struct `mln::accu::shape::height< I >`
Height accumulator.
- struct `mln::accu::shape::volume< I >`
Volume accumulator class.

8.2.1 Detailed Description

Accumulators working on images.

8.3 On values

Accumulators working on image values.

Classes

- struct `mln::accu::convolve< T1, T2, R >`
Generic convolution accumulator class.
- struct `mln::accu::count_labels< L >`
Count the number of different labels in an image.
- struct `mln::accu::count_value< V >`
Define an accumulator that counts the occurrence of a given value.
- struct `mln::accu::histo< V >`
Generic histogram class over a value set with type V.
- struct `mln::accu::label_used< L >`
References all the labels used.
- struct `mln::accu::logic::land`
"Logical-and" accumulator.
- struct `mln::accu::logic::land_basic`
"Logical-and" accumulator.
- struct `mln::accu::logic::lor`
"Logical-or" accumulator.
- struct `mln::accu::logic::lor_basic`
"Logical-or" accumulator class.

- struct `mln::accu::maj_h< T >`
Compute the majority value.
- struct `mln::accu::math::inf< T >`
Generic inf accumulator class.
- struct `mln::accu::math::sum< T, S >`
Generic sum accumulator class.
- struct `mln::accu::math::sup< T >`
Generic sup accumulator class.
- struct `mln::accu::rms< T, V >`
Generic root mean square accumulator class.
- struct `mln::accu::stat::deviation< T, S, M >`
Generic standard deviation accumulator class.
- struct `mln::accu::stat::histo3d_rgb< V >`
Define a histogram as accumulator which returns an `image3d`.
- struct `mln::accu::stat::max< T >`
Generic max accumulator class.
- struct `mln::accu::stat::max_h< V >`
Generic max function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::mean< T, S, M >`
Generic mean accumulator class.
- struct `mln::accu::stat::median_alt< S >`
Generic `median_alt` function based on histogram over a value set with type `S`.
- struct `mln::accu::stat::median_h< V >`
Generic median function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::min< T >`
Generic min accumulator class.
- struct `mln::accu::stat::min_h< V >`
Generic min function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::min_max< V >`
Generic min and max accumulator class.
- struct `mln::accu::stat::rank< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::rank< bool >`
rank accumulator class for Boolean.

- struct `mln::accu::stat::rank_high_quant< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::var< T >`
Var accumulator class.
- struct `mln::accu::stat::variance< T, S, R >`
Variance accumulator class.

8.3.1 Detailed Description

Accumulators working on image values.

8.4 Multiple accumulators

Set of special accumulators for computing several accumulators at the same time.

Classes

- struct `mln::accu::pair< A1, A2, T >`
Generic pair of accumulators.
- struct `mln::accu::tuple< A, n, >`
Generic tuple of accumulators.

8.4.1 Detailed Description

Set of special accumulators for computing several accumulators at the same time.

8.5 Graphes

All graphes implementations.

Classes

- class `mln::util::graph`
Undirected graph.
- class `mln::util::line_graph< G >`
Undirected line graph of a graph of type G.

8.5.1 Detailed Description

All graphes implementations.

8.6 Images

All the generic image types provided in Olena.

Modules

- [Basic types](#)
Concrete images.
- [Image morphers](#)
Morpher on both image values and domain.
- [Values morphers](#)
Morpher on image values.
- [Domain morphers](#)
Morpher on image domain.
- [Identity morphers](#)
Morpher adding new fonctionnalités.

8.6.1 Detailed Description

All the generic image types provided in Olena.

8.7 Basic types

Concrete images.

Classes

- class [mln::complex_image< D, G, V >](#)
Image based on a complex.
- class [mln::edge_image< P, V, G >](#)
Image based on graph edges.
- struct [mln::flat_image< T, S >](#)
Image with a single value.
- struct [mln::image1d< T >](#)

Basic 1D image class.

- class `mln::image2d< T >`
Basic 2D image class.
- struct `mln::image2d_h< V >`
2d image based on an hexagonal mesh.
- struct `mln::image3d< T >`
Basic 3D image class.
- class `mln::pw::image< F, S >`
A generic point-wise image implementation.
- class `mln::vertex_image< P, V, G >`
Image based on graph vertices.

8.7.1 Detailed Description

Concrete images.

8.8 Image morphers

Morpher on both image values and domain.

Morpher on both image values and domain.

8.9 Values morphers

Morpher on image values.

Classes

- struct `mln::fun_image< F, I >`
Image read through a function.
- class `mln::thru_image< I, F >`
Morph image values through a function.
- class `mln::thrubin_image< I1, I2, F >`
Morphes values from two images through a binary function.
- struct `mln::violent_cast_image< T, I >`
Violently cast image values to a given type.

8.9.1 Detailed Description

Morpher on image values.

8.10 Domain morphers

Morpher on image domain.

Classes

- struct `mln::extended< I >`
Makes an image become restricted by a point set.
- class `mln::extension_fun< I, F >`
Extends the domain of an image with a function.
- class `mln::extension_ima< I, J >`
Extends the domain of an image with an image.
- class `mln::extension_val< I >`
Extends the domain of an image with a value.
- struct `mln::hexa< I >`
hexagonal image class.
- struct `mln::p2p_image< I, F >`
FIXME: Doc!
- class `mln::sub_image< I, S >`
Image having its domain restricted by a site set.
- struct `mln::sub_image_if< I, S >`
Image having its domain restricted by a site set and a function.
- struct `mln::transformed_image< I, F >`
Image having its domain restricted by a site set.
- struct `mln::unproject_image< I, D, F >`
Un-projects an image.

8.10.1 Detailed Description

Morpher on image domain.

8.11 Identity morphers

Morpher adding new fonctionnalities.

Classes

- struct [mln::decorated_image< I, D >](#)
Image that can have additional features.
- class [mln::labeled_image< I >](#)
Morpher providing an improved interface for labeled image.
- struct [mln::lazy_image< I, F, B >](#)
Image values are computed on the fly.
- class [mln::plain< I >](#)
Prevents an image from sharing its data.
- class [mln::safe_image< I >](#)
Makes an image accessible at undefined location.
- struct [mln::tr_image< S, I, T >](#)
Transform an image by a given transformation.

8.11.1 Detailed Description

Morpher adding new fonctionnalités.

8.12 Types

Milena Object types.

Modules

- [Graphes](#)
All graphes implementations.
- [Images](#)
All the generic image types provided in Olena.
- [Neighborhoods](#)
All the predefined generic neighborhoods.
- [Site sets](#)
All Site set types.
- [Utilities](#)
Miscellaneous useful containers/structures.
- [Windows](#)
All the predefined generic windows.

8.12.1 Detailed Description

Milena Object types.

8.13 Accumulators

All accumulator types.

Modules

- [On site sets](#)

Accumulators working on site sets.

- [On images](#)

Accumulators working on images.

- [On values](#)

Accumulators working on image values.

- [Multiple accumulators](#)

Set of special accumulators for computing several accumulators at the same time.

8.13.1 Detailed Description

All accumulator types.

8.14 Routines

All algorithms/routines provided in Milena.

All algorithms/routines provided in Milena.

8.15 Canvas

All canvas.

All canvas.

8.16 Functions

All predefined functions.

Classes

- struct [mln::Function< E >](#)
Base class for implementation of function-objects.
- struct [mln::Function_n2v< E >](#)
Base class for implementation of function-objects from Nil to value.
- struct [mln::Function_v2b< E >](#)
Base class for implementation of function-objects from a value to a Boolean.
- struct [mln::Function_v2v< E >](#)
Base class for implementation of function-objects from value to value.
- struct [mln::Function_vv2b< E >](#)
Base class for implementation of function-objects from a couple of values to a Boolean.
- struct [mln::Function_vv2v< E >](#)
Base class for implementation of function-objects from a couple of values to a value.

Namespaces

- namespace [mln::fun::i2v](#)
Namespace of integer-to-value functions.
- namespace [mln::fun::n2v](#)
Namespace of functions from nil to value.
- namespace [mln::fun::stat](#)
Namespace of statistical functions.
- namespace [mln::fun::v2i](#)
Namespace of value-to-integer functions.
- namespace [mln::fun::v2v](#)
Namespace of functions from value to value.

Modules

- [v2w2v functions](#)
All bijective functions.
- [v2w_w2v functions](#)
All bijective function.
- [vv2b functions](#)
All functions mapping two values to a logical value.

8.16.1 Detailed Description

All predefined functions.

8.17 Neighborhoods

All the predefined generic neighborhoods.

Modules

- [1D neighborhoods](#)
Predefined 1D neighborhoods.
- [2D neighborhoods](#)
Predefined 2D neighborhoods.
- [3D neighborhoods](#)
Predefined 3D neighborhoods.

8.17.1 Detailed Description

All the predefined generic neighborhoods.

8.18 1D neighborhoods

Predefined 1D neighborhoods.

Typedefs

- `typedef neighb< window1d > mln::neighb1d`
Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

Functions

- `const neighb1d & mln::c2 ()`
2-connectivity neighborhood on the 1D grid.

8.18.1 Detailed Description

Predefined 1D neighborhoods.

8.18.2 Typedef Documentation

8.18.2.1 typedef neighb<window1d> mln::neighb1d

Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

Definition at line 47 of file neighb1d.hh.

8.18.3 Function Documentation

8.18.3.1 const neighb1d & mln::c2() [inline]

2-connectivity neighborhood on the 1D grid.

○ × ○

Returns

A neighb1d.

Definition at line 67 of file neighb1d.hh.

8.19 2D neighborhoods

Predefined 2D neighborhoods.

Typedefs

- typedef neighb< window2d > [mln::neighb2d](#)
Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

Functions

- const neighb2d & [mln::c2_col](#) ()
Vertical 2-connectivity neighborhood on the 2D grid.
- const neighb2d & [mln::c2_row](#) ()
Horizontal 2-connectivity neighborhood on the 2D grid.
- const neighb2d & [mln::c4](#) ()
4-connectivity neighborhood on the 2D grid.
- const neighb2d & [mln::c8](#) ()
8-connectivity neighborhood on the 2D grid.

8.19.1 Detailed Description

Predefined 2D neighborhoods.

8.19.2 Typedef Documentation

8.19.2.1 typedef neighb<window2d> mln::neighb2d

Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

Definition at line 51 of file core/alias/neighb2d.hh.

8.19.3 Function Documentation

8.19.3.1 const neighb2d & mln::c2_col () [inline]

Vertical 2-connectivity neighborhood on the 2D grid.

```
- o -
- x -
- o -
```

Returns

A neighb2d.

Definition at line 190 of file core/alias/neighb2d.hh.

8.19.3.2 const neighb2d & mln::c2_row () [inline]

Horizontal 2-connectivity neighborhood on the 2D grid.

```
- - -
o x o
- - -
```

Returns

A neighb2d.

Definition at line 176 of file core/alias/neighb2d.hh.

8.19.3.3 const neighb2d & mln::c4 () [inline]

4-connectivity neighborhood on the 2D grid.

```
- o -
o x o
- o -
```

Returns

A neighb2d.

Definition at line 148 of file core/alias/neighb2d.hh.

8.19.3.4 const neighb2d & mln::c8 () [inline]

8-connectivity neighborhood on the 2D grid.

```

○ ○ ○
○ × ○
○ ○ ○

```

Returns

A neighb2d.

Definition at line 162 of file core/alias/neighb2d.hh.

8.20 3D neighborhoods

Predefined 3D neighborhoods.

Typedefs

- typedef neighb< window3d > [mln::neighb3d](#)
Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

Functions

- const neighb3d & [mln::c18](#) ()
18-connectivity neighborhood on the 3D grid.
- const neighb3d & [mln::c26](#) ()
26-connectivity neighborhood on the 3D grid.
- const neighb3d & [mln::c2_3d_sli](#) ()
depth 2-connectivity neighborhood on the 3D grid.
- const neighb3d & [mln::c4_3d](#) ()
4-connectivity neighborhood on the 3D grid.
- const neighb3d & [mln::c6](#) ()
6-connectivity neighborhood on the 3D grid.
- const neighb3d & [mln::c8_3d](#) ()
8-connectivity neighborhood on the 3D grid.

8.20.1 Detailed Description

Predefined 3D neighborhoods.

8.20.2 Typedef Documentation

8.20.2.1 `typedef neighb<window3d> mln::neighb3d`

Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

Definition at line 50 of file `neighb3d.hh`.

8.20.3 Function Documentation

8.20.3.1 `const neighb3d & mln::c18 () [inline]`

18-connectivity neighborhood on the 3D grid.

```

      . o .
    o o o
    . o .

      o o o
    o x o
    o o o

      . o .
    o o o
    . o .

```

Returns

A `neighb3d`.

Definition at line 288 of file `neighb3d.hh`.

References `mln::c6()`, `mln::window< D >::insert()`, and `mln::win::sym()`.

Referenced by `mln::c26()`.

8.20.3.2 `const neighb3d & mln::c26 () [inline]`

26-connectivity neighborhood on the 3D grid.

```

      o o o
    o o o
    o o o

      o o o
    o x o
    o o o

      o o o
    o o o

```

○ ○ ○

Returns

A neighb3d.

Definition at line 309 of file neighb3d.hh.

References mln::c18(), mln::window< D >::insert(), and mln::win::sym().

8.20.3.3 const neighb3d & mln::c2_3d_sli() [inline]

depth 2-connectivity neighborhood on the 3D grid.

```

      . . .
      . ○ .
      . . .

      . . .
      . x .
      . . .

      . . .
      . ○ .
      . . .

```

Returns

A neighb3d.

Definition at line 226 of file neighb3d.hh.

References mln::window< D >::insert().

8.20.3.4 const neighb3d & mln::c4_3d() [inline]

4-connectivity neighborhood on the 3D grid.

```

      . . .
      . . .
      . . .

      . ○ .
      ○ x ○
      . ○ .

      . . .
      . . .
      . . .

```

Returns

A neighb3d.

Definition at line 241 of file neighb3d.hh.

References mln::window< D >::insert(), and mln::win::sym().

8.20.3.5 const neighb3d & mln::c6 () [inline]

6-connectivity neighborhood on the 3D grid.

```

      . . .
      . o .
      . . .

      . o .
      o x o
      . o .

      . . .
      . o .
      . . .

```

Returns

A neighb3d.

Definition at line 271 of file neighb3d.hh.

References mln::window< D >::insert(), and mln::win::sym().

Referenced by mln::c18().

8.20.3.6 const neighb3d & mln::c8_3d () [inline]

8-connectivity neighborhood on the 3D grid.

```

      . . .
      . . .
      . . .

      o o o
      o x o
      o o o

      . . .
      . . .
      . . .

```

Returns

A neighb3d.

Definition at line 257 of file neighb3d.hh.

8.21 Site sets

All Site set types.

Modules

- [Basic types](#)
Basic site sets.
- [Graph based](#)
Site sets based on a graph.
- [Complex based](#)
Site sets based on a complexes.
- [Sparse types](#)
Sparse site sets.
- [Queue based](#)
Site sets based on a queue.

8.21.1 Detailed Description

All Site set types.

8.22 Basic types

Basic site sets.

Classes

- class [mln::box< P >](#)
Generic box class: site set containing points of a regular grid.
- class [mln::p_line2d](#)
2D discrete line of points.
- class [mln::p_mutable_array_of< S >](#)
[p_mutable_array_of](#) is a mutable array of site sets.
- class [mln::p_run< P >](#)
[Point](#) set class in run.

8.22.1 Detailed Description

Basic site sets.

8.23 Graph based

Site sets based on a graph.

Classes

- class `mln::p_edges< G, F >`
Site set mapping graph edges and image sites.
- struct `mln::p_faces< N, D, P >`
A complex psite set based on the N-faces of a complex of dimension D (a D-complex).
- class `mln::p_vertices< G, F >`
Site set based mapping graph vertices to sites.

8.23.1 Detailed Description

Site sets based on a graph.

8.24 Complex based

Site sets based on a complexes.

Classes

- class `mln::p_complex< D, G >`
A complex psite set based on the N-faces of a complex of dimension D (a D-complex).

8.24.1 Detailed Description

Site sets based on a complexes.

8.25 Sparse types

Sparse site sets.

Classes

- class `mln::p_array< P >`
Multi-set of sites.
- class `mln::p_centered< W >`
Site set corresponding to a window centered on a site.

- class `mln::p_if< S, F >`
Site set restricted w.r.t.
- class `mln::p_image< I >`
Site set based on an image of Booleans.
- class `mln::p_set< P >`
Mathematical set of sites (based on `util::set`).
- class `mln::p_transformed< S, F >`
Site set transformed through a function.
- class `mln::p_vaccess< V, S >`
Site set in which sites are grouped by their associated value.

8.25.1 Detailed Description

Sparse site sets.

8.26 Queue based

Site sets based on a queue.

Classes

- class `mln::p_key< K, P >`
Priority queue class.
- class `mln::p_priority< P, Q >`
Priority queue.
- class `mln::p_queue< P >`
Queue of sites (based on `std::deque`).
- class `mln::p_queue_fast< P >`
Queue of sites class (based on `p_array`).

8.26.1 Detailed Description

Site sets based on a queue.

8.27 Utilities

Miscellaneous useful containers/structures.

Classes

- class `mln::util::adjacency_matrix< V >`
A class of adjacency matrix.
- class `mln::util::array< T >`
A dynamic array class.
- class `mln::util::couple< T, U >`
Definition of a couple.
- struct `mln::util::eat`
Eat structure.
- class `mln::util::fibonacci_heap< P, T >`
Fibonacci heap.
- struct `mln::util::ignore`
Ignore structure.
- struct `mln::util::nil`
Nil structure.
- struct `mln::util::ord_pair< T >`
Ordered pair structure s.a.
- class `mln::util::set< T >`
An "efficient" mathematical set class.
- class `mln::util::site_pair< P >`
A pair of sites.
- class `mln::util::soft_heap< T, R >`
Soft heap.
- struct `mln::util::tracked_ptr< T >`
Smart pointer for shared data with tracking.
- struct `mln::util::yes`
Object that always says "yes".

8.27.1 Detailed Description

Miscellaneous useful containers/structures.

8.28 Windows

All the predefined generic windows.

Modules

- [1D windows](#)
Predefined 1D windows.
- [2D windows](#)
Predefined 2D windows.
- [3D windows](#)
Predefined 3D windows.
- [N-D windows](#)
Predefined N-D windows.
- [Multiple windows](#)
Generic multiple windows.

8.28.1 Detailed Description

All the predefined generic windows.

8.29 1D windows

Predefined 1D windows.

Typedefs

- typedef line< grid::tick, 0, def::coord > [mln::win::segment1d](#)
Segment window defined on the 1D grid.
- typedef window< [mln::dpoint1d](#) > [mln::window1d](#)
Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.

8.29.1 Detailed Description

Predefined 1D windows.

8.29.2 Typedef Documentation

8.29.2.1 typedef line<grid::tick, 0, def::coord> mln::win::segment1d

Segment window defined on the 1D grid.

An segment1d is centered and symmetric; so its height (length) is odd.

For instance:

○ × ○

is defined with length = 3.

Definition at line 56 of file segment1d.hh.

8.29.2.2 typedef window<mln::dpoint1d> mln::window1d

Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.

Definition at line 47 of file window1d.hh.

8.30 2D windows

Predefined 2D windows.

Classes

- struct [mln::win::backdiag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [mln::win::diag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [mln::win::octagon2d](#)
Octagon window defined on the 2D square grid.
- struct [mln::win::rectangle2d](#)
Rectangular window defined on the 2D square grid.

Typedefs

- typedef ball< grid::square, def::coord > [mln::win::disk2d](#)
2D disk window; precisely, ball-shaped window defined on the 2D square grid.
- typedef line< grid::square, 1, def::coord > [mln::win::hline2d](#)
Horizontal line window defined on the 2D square grid.
- typedef line< grid::square, 0, def::coord > [mln::win::vline2d](#)
Vertical line window defined on the 2D square grid.
- typedef window< [mln::dpoint2d](#) > [mln::window2d](#)
Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.

Functions

- `const window2d & mln::win_c4p ()`
4-connectivity window on the 2D grid, including the center.
- `const window2d & mln::win_c8p ()`
8-connectivity window on the 2D grid, including the center.

8.30.1 Detailed Description

Predefined 2D windows.

8.30.2 Typedef Documentation

8.30.2.1 `typedef ball<grid::square, def::coord> mln::win::disk2d`

2D disk window; precisely, ball-shaped window defined on the 2D square grid.

Definition at line 49 of file `disk2d.hh`.

8.30.2.2 `typedef line<grid::square, 1, def::coord> mln::win::hline2d`

Horizontal line window defined on the 2D square grid.

An `hline2d` is centered and symmetric; so its height is 1 and its width (length) is odd.

For instance:

```
○ ○ x ○ ○
```

is defined with `length = 5`.

Definition at line 57 of file `hline2d.hh`.

8.30.2.3 `typedef line<grid::square, 0, def::coord> mln::win::vline2d`

Vertical line window defined on the 2D square grid.

An `vline2d` is centered and symmetric; so its width is 1 and its height (length) is odd.

For instance:

```
○
x
○
```

is defined with `length = 3`.

Definition at line 58 of file `vline2d.hh`.

8.30.2.4 `typedef window<mln::dpoint2d> mln::window2d`

Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.

Definition at line 49 of file `window2d.hh`.

8.30.3 Function Documentation

8.30.3.1 `const window2d & mln::win_c4p () [inline]`

4-connectivity window on the 2D grid, including the center.

```

-  o  -
o  x  o
-  o  -

```

Returns

A `window2d`.

Definition at line 105 of file `window2d.hh`.

References `mln::window< D >::insert()`, and `mln::window< D >::size()`.

8.30.3.2 `const window2d & mln::win_c8p () [inline]`

8-connectivity window on the 2D grid, including the center.

```

o  o  o
o  x  o
o  o  o

```

Returns

A `window2d`.

Definition at line 121 of file `window2d.hh`.

References `mln::window< D >::insert()`, and `mln::window< D >::size()`.

8.31 3D windows

Predefined 3D windows.

Classes

- struct [mln::win::cube3d](#)
Cube window defined on the 3D grid.
- struct [mln::win::cuboid3d](#)
Cuboid defined on the 3-D square grid.

Typedefs

- typedef line< grid::cube, 0, def::coord > [mln::win::sline3d](#)
Depth line window defined on the 3D cubic grid.
- typedef ball< grid::cube, def::coord > [mln::win::sphere3d](#)
3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.
- typedef window< [mln::dpoint3d](#) > [mln::window3d](#)
Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.

Functions

- const window3d & [mln::win_c4p_3d](#) ()
4-connectivity window on the 3D grid, including the center.
- const window3d & [mln::win_c8p_3d](#) ()
8-connectivity window on the 3D grid, including the center.

8.31.1 Detailed Description

Predefined 3D windows.

8.31.2 Typedef Documentation

8.31.2.1 typedef line<grid::cube, 0, def::coord> mln::win::sline3d

Depth line window defined on the 3D cubic grid.

An sline3d is centered and symmetric; so its height and its width are 1 and its depth is odd.

For instance:

```

      . . .
      . o .
      . . .

      . . .
      . x .
      . . .

      . . .
      . o .
      . . .

```

is defined with length = 3.

Definition at line 68 of file sline3d.hh.

8.31.2.2 typedef ball<grid::cube, def::coord> mln::win::sphere3d

3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.

Definition at line 47 of file sphere3d.hh.

8.31.2.3 typedef window<mln::dpoint3d> mln::window3d

Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.

Definition at line 48 of file window3d.hh.

8.31.3 Function Documentation**8.31.3.1 const window3d & mln::win_c4p_3d () [inline]**

4-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -

  - o -
  o x o
- o -

  - - -
  - - -
  - - -

```

Returns

A window3d.

Definition at line 119 of file window3d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

8.31.3.2 const window3d & mln::win_c8p_3d () [inline]

8-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -

  o o o
  o x o
  o o o

  - - -
  - - -
  - - -

```

Returns

A window3d.

Definition at line 135 of file window3d.hh.

References `mln::window< D >::insert()`, and `mln::window< D >::size()`.

8.32 N-D windows

Predefined N-D windows.

Classes

- struct `mln::win::ball< G, C >`
Generic ball window defined on a given grid.
- struct `mln::win::line< M, i, C >`
Generic line window defined on a given grid in the given dimension.

8.32.1 Detailed Description

Predefined N-D windows.

8.33 Multiple windows

Generic multiple windows.

Classes

- class `mln::win::multiple< W, F >`
Multiple window.
- class `mln::win::multiple_size< n, W, F >`
Definition of a multiple-size window.

8.33.1 Detailed Description

Generic multiple windows.

8.34 v2w2v functions

All bijective functions.

All bijective functions.

8.35 v2w_w2v functions

All bijective function.

All bijective function.

8.36 vv2b functions

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.

Chapter 9

Namespace Documentation

9.1 mln Namespace Reference

[mln/convert/to_image.hh](#)

Namespaces

- namespace [accu](#)
Namespace of accumulators.
- namespace [algebra](#)
Namespace of algebraic structure.
- namespace [arith](#)
Namespace of arithmetic.
- namespace [binarization](#)
Namespace of "point-wise" expression tools.
- namespace [border](#)
Namespace of routines related to image virtual (outer) border.
- namespace [canvas](#)
Namespace of canvas.
- namespace [convert](#)
Namespace of conversion routines.
- namespace [data](#)
Namespace of image processing routines related to pixel data.
- namespace [debug](#)
Namespace of routines that help to debug.
- namespace [def](#)

Namespace for core definitions.

- namespace [display](#)

Namespace of routines that help to display images.

- namespace [doc](#)

The namespace [mln::doc](#) is only for documentation purpose.

- namespace [draw](#)

Namespace of drawing routines.

- namespace [estim](#)

Namespace of estimation materials.

- namespace [extension](#)

Namespace of extension tools.

- namespace [fun](#)

Namespace of functions.

- namespace [geom](#)

Namespace of all things related to geometry.

- namespace [graph](#)

Namespace of graph related routines.

- namespace [grid](#)

Namespace of grids definitions.

- namespace [histo](#)

Namespace of histograms.

- namespace [impl](#)

Implementation namespace of mln namespace.

- namespace [io](#)

Namespace of input/output handling.

- namespace [labeling](#)

Namespace of labeling routines.

- namespace [linear](#)

Namespace of linear image processing routines.

- namespace [literal](#)

Namespace of literals.

- namespace [logical](#)

Namespace of logic.

- namespace [make](#)
Namespace of routines that help to make Milena's objects.
- namespace [math](#)
Namespace of mathematical routines.
- namespace [metal](#)
Namespace of meta-programming tools.
- namespace [morpho](#)
Namespace of mathematical morphology routines.
- namespace [norm](#)
Namespace of norms.
- namespace [opt](#)
Namespace of optional routines.
- namespace [pw](#)
Namespace of "point-wise" expression tools.
- namespace [registration](#)
Namespace of "point-wise" expression tools.
- namespace [select](#)
Select namespace (FIXME doc).
- namespace [set](#)
Namespace of image processing routines related to pixel sets.
- namespace [subsampling](#)
Namespace of "point-wise" expression tools.
- namespace [tag](#)
Namespace of image processing routines related to tags.
- namespace [test](#)
Namespace of image processing routines related to pixel tests.
- namespace [topo](#)
Namespace of "point-wise" expression tools.
- namespace [trace](#)
Namespace of routines related to the trace mechanism.
- namespace [trait](#)
Namespace where traits are defined.
- namespace [transform](#)
Namespace of transforms.

- namespace [util](#)
Namespace of tools using for more complex algorithm.
- namespace [value](#)
Namespace of materials related to pixel value types.
- namespace [win](#)
Namespace of image processing routines related to win.

Classes

- struct [Accumulator](#)
Base class for implementation of accumulators.
- class [bkd_pixter1d](#)
Backward pixel iterator on a 1-D image with border.
- class [bkd_pixter2d](#)
Backward pixel iterator on a 2-D image with border.
- class [bkd_pixter3d](#)
Backward pixel iterator on a 3-D image with border.
- class [box](#)
Generic box class: site set containing points of a regular grid.
- struct [Box](#)
Base class for implementation classes of boxes.
- class [box_runend_piter](#)
A generic backward iterator on points by lines.
- class [box_runstart_piter](#)
A generic forward iterator on points by lines.
- struct [Browsing](#)
Base class for implementation classes that are browsings.
- struct [category< R\(*\) \(A\) >](#)
Category declaration for a unary C function.
- class [complex_image](#)
Image based on a complex.
- class [complex_neighborhood_bkd_piter](#)
Backward iterator on complex neighborhood.

- class [complex_neighborhood_fwd_piter](#)
Forward iterator on complex neighborhood.
- class [complex_psite](#)
Point site associated to a `mln::p_complex`.
- class [complex_window_bkd_piter](#)
Backward iterator on complex window.
- class [complex_window_fwd_piter](#)
Forward iterator on complex window.
- struct [decorated_image](#)
Image that can have additional features.
- struct [Delta_Point_Site](#)
FIXME: Doc!
- struct [Delta_Point_Site< void >](#)
Delta point site category flag type.
- struct [dpoint](#)
Generic delta-point class.
- struct [Dpoint](#)
Base class for implementation of delta-point classes.
- class [dpoints_bkd_pixter](#)
A generic backward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpoints_fwd_pixter](#)
A generic forward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpsites_bkd_piter](#)
A generic backward iterator on points of windows and of neighborhoods.
- class [dpsites_fwd_piter](#)
A generic forward iterator on points of windows and of neighborhoods.
- struct [Edge](#)
edge category flag type.
- class [edge_image](#)
Image based on graph edges.
- struct [extended](#)
Makes an image become restricted by a point set.
- class [extension_fun](#)
Extends the domain of an image with a function.

- class [extension_ima](#)
Extends the domain of an image with an image.
- class [extension_val](#)
Extends the domain of an image with a value.
- struct [flat_image](#)
Image with a single value.
- struct [fun_image](#)
Image read through a function.
- struct [Function](#)
Base class for implementation of function-objects.
- struct [Function< void >](#)
Function category flag type.
- struct [Function_n2v](#)
Base class for implementation of function-objects from Nil to value.
- struct [Function_v2b](#)
Base class for implementation of function-objects from a value to a Boolean.
- struct [Function_v2v](#)
Base class for implementation of function-objects from value to value.
- struct [Function_vv2b](#)
Base class for implementation of function-objects from a couple of values to a Boolean.
- struct [Function_vv2v](#)
Base class for implementation of function-objects from a couple of values to a value.
- class [fwd_pixter1d](#)
Forward pixel iterator on a 1-D image with border.
- class [fwd_pixter2d](#)
Forward pixel iterator on a 2-D image with border.
- class [fwd_pixter3d](#)
Forward pixel iterator on a 3-D image with border.
- struct [Gdpoint](#)
FIXME: Doc!
- struct [Gdpoint< void >](#)
Delta point site category flag type.
- struct [Generalized_Pixel](#)

Base class for implementation classes that are pixels or that have the behavior of pixels.

- struct [Gpoint](#)

Base class for implementation of point classes.

- struct [Graph](#)

Base class for implementation of graph classes.

- struct [graph_elt_mixed_neighborhood](#)

Elementary neighborhood on graph class.

- class [graph_elt_mixed_window](#)

Elementary window on graph class.

- struct [graph_elt_neighborhood](#)

Elementary neighborhood on graph class.

- struct [graph_elt_neighborhood_if](#)

Elementary neighborhood_if on graph class.

- class [graph_elt_window](#)

Elementary window on graph class.

- class [graph_elt_window_if](#)

Custom window on graph class.

- class [graph_window_base](#)

- class [graph_window_if_piter](#)

Forward iterator on line graph window.

- class [graph_window_piter](#)

Forward iterator on line graph window.

- struct [hexa](#)

hexagonal image class.

- struct [Image](#)

Base class for implementation of image classes.

- struct [image1d](#)

Basic 1D image class.

- class [image2d](#)

Basic 2D image class.

- struct [image2d_h](#)

2d image based on an hexagonal mesh.

- struct [image3d](#)

Basic 3D image class.

- struct [interpolated](#)
Makes the underlying image being accessed with floating coordinates.
- struct [Iterator](#)
Base class for implementation classes that are iterators.
- class [labeled_image](#)
Morpher providing an improved interface for labeled image.
- class [labeled_image_base](#)
Base class Morpher providing an improved interface for labeled image.
- struct [lazy_image](#)
Image values are computed on the fly.
- struct [Literal](#)
Base class for implementation classes of literals.
- struct [Mesh](#)
Base class for implementation classes of meshes.
- struct [Meta_Accumulator](#)
Base class for implementation of meta accumulators.
- struct [Meta_Function](#)
Base class for implementation of meta functions.
- struct [Meta_Function_v2v](#)
Base class for implementation of function-objects from value to value.
- struct [Meta_Function_vv2v](#)
Base class for implementation of function-objects from value to value.
- class [neighb](#)
Adapter class from window to neighborhood.
- struct [Neighborhood](#)
Base class for implementation classes that are neighborhoods.
- struct [Neighborhood< void >](#)
Neighborhood category flag type.
- struct [Object](#)
Base class for almost every class defined in Milena.
- struct [p2p_image](#)
FIXME: Doc!
- class [p_array](#)

Multi-set of sites.

- class [p_centered](#)
Site set corresponding to a window centered on a site.
- class [p_complex](#)
A complex psite set based on the N-faces of a complex of dimension D (a `D-complex`).
- class [p_edges](#)
Site set mapping graph edges and image sites.
- struct [p_faces](#)
A complex psite set based on the N-faces of a complex of dimension D (a `D-complex`).
- class [p_graph_piter](#)
Generic iterator on point sites of a `mln::S`.
- class [p_if](#)
Site set restricted w.r.t.
- class [p_image](#)
Site set based on an image of Booleans.
- class [p_indexed_bkd_piter](#)
Backward iterator on sites of an indexed site set.
- class [p_indexed_fwd_piter](#)
Forward iterator on sites of an indexed site set.
- class [p_indexed_psite](#)
Psite class for indexed site sets such as [p_array](#).
- class [p_key](#)
Priority queue class.
- class [p_line2d](#)
2D discrete line of points.
- class [p_mutable_array_of](#)
[p_mutable_array_of](#) is a mutable array of site sets.
- class [p_n_faces_bkd_piter](#)
Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.
- class [p_n_faces_fwd_piter](#)
Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.
- class [p_priority](#)
Priority queue.

- class [p_queue](#)
Queue of sites (based on `std::deque`).
- class [p_queue_fast](#)
Queue of sites class (based on [p_array](#)).
- class [p_run](#)
Point set class in run.
- class [p_set](#)
Mathematical set of sites (based on [util::set](#)).
- class [p_transformed](#)
Site set transformed through a function.
- struct [p_transformed_piter](#)
Iterator on `p_transformed<S,F>`.
- class [p_vaccess](#)
Site set in which sites are grouped by their associated value.
- class [p_vertices](#)
Site set based mapping graph vertices to sites.
- struct [pixel](#)
Generic pixel class.
- class [plain](#)
Prevents an image from sharing its data.
- struct [point](#)
Generic point class.
- struct [Point](#)
Base class for implementation of point classes.
- struct [Proxy](#)
Base class for implementation classes of the notion of "proxy".
- struct [Proxy< void >](#)
Proxy category flag type.
- struct [Pseudo_Site](#)
Base class for implementation classes of the notion of "pseudo site".
- struct [Pseudo_Site< void >](#)
Pseudo_Site category flag type.
- struct [Regular_Grid](#)
Base class for implementation classes of regular grids.

- class [safe_image](#)
Makes an image accessible at undefined location.
- struct [Site](#)
Base class for classes that are explicitly sites.
- struct [Site< void >](#)
[Site](#) category flag type.
- struct [Site_Iterator](#)
Base class for implementation of classes of iterator on points.
- struct [Site_Proxy](#)
Base class for implementation classes of the notion of "site proxy".
- struct [Site_Proxy< void >](#)
[Site_Proxy](#) category flag type.
- struct [Site_Set](#)
Base class for implementation classes of site sets.
- struct [Site_Set< void >](#)
[Site_Set](#) category flag type.
- class [sub_image](#)
[Image](#) having its domain restricted by a site set.
- struct [sub_image_if](#)
[Image](#) having its domain restricted by a site set and a function.
- class [thru_image](#)
Morph image values through a function.
- class [thrubin_image](#)
Morphes values from two images through a binary function.
- struct [tr_image](#)
Transform an image by a given transformation.
- struct [transformed_image](#)
[Image](#) having its domain restricted by a site set.
- struct [unproject_image](#)
Un-projects an image.
- struct [Value](#)
Base class for implementation classes of values.
- struct [Vertex](#)

Vertex category flag type.

- class `vertex_image`
Image based on graph vertices.
- struct `violent_cast_image`
Violently cast image values to a given type.
- struct `w_window`
Generic `w_window` class.
- struct `Weighted_Window`
Base class for implementation classes that are `weighted_windows`.
- class `window`
Generic window class.
- struct `Window`
Base class for implementation classes that are windows.

Typedefs

- typedef `mln::complex_image< 1, mln::discrete_plane_1complex_geometry, bool > bin_1complex_image2d`
Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef `mln::complex_image< 2, mln::space_2complex_geometry, bool > bin_2complex_image3df`
Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `box< mln::point1d > box1d`
Type alias for a box defined on the 1D square grid with integer coordinates.
- typedef `box< mln::point2d > box2d`
Type alias for a box defined on the 2D square grid with integer coordinates.
- typedef `box< point2d_h > box2d_h`
FIXME.
- typedef `box< point3d > box3d`
Type alias for a box defined on the 3D square grid with integer coordinates.
- typedef `mln::geom::complex_geometry< 1, point2d > discrete_plane_1complex_geometry`
Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).
- typedef `mln::geom::complex_geometry< 2, point2d > discrete_plane_2complex_geometry`

Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).

- typedef [dpoint](#)< [mln::grid::tick](#), [def::coord](#) > [dpoint1d](#)
Type alias for a delta-point defined on the 1D square grid with integer coordinates.
- typedef [dpoint](#)< [mln::grid::square](#), [mln::def::coord](#) > [dpoint2d](#)
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef [dpoint](#)< [mln::grid::hexa](#), [def::coord](#) > [dpoint2d_h](#)
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef [dpoint](#)< [mln::grid::cube](#), [def::coord](#) > [dpoint3d](#)
Type alias for a delta-point defined on the 3D square grid with integer coordinates.
- typedef [mln::complex_image](#)< 2, [mln::space_2complex_geometry](#), [float](#) > [float_2complex_image3df](#)
Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef [mln::complex_image](#)< 1, [mln::discrete_plane_1complex_geometry](#), [mln::value::int_u8](#) > [int_u8_1complex_image2d](#)
Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef [mln::complex_image](#)< 2, [mln::discrete_plane_2complex_geometry](#), [mln::value::int_u8](#) > [int_u8_2complex_image2d](#)
Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef [mln::complex_image](#)< 2, [mln::space_2complex_geometry](#), [mln::value::int_u8](#) > [int_u8_2complex_image3df](#)
Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef [neighb](#)< [window1d](#) > [neighb1d](#)
Type alias for a neighborhood defined on the 1D square grid with integer coordinates.
- typedef [neighb](#)< [window2d](#) > [neighb2d](#)
Type alias for a neighborhood defined on the 2D square grid with integer coordinates.
- typedef [neighb](#)< [window3d](#) > [neighb3d](#)
Type alias for a neighborhood defined on the 3D square grid with integer coordinates.
- typedef [p_run](#)< [point2d](#) > [p_run2d](#)
Type alias for a run of 2d points.
- typedef [p_set_of](#)< [p_run2d](#) > [p_runs2d](#)
Type alias for a set of runs of 2d points.
- typedef [point](#)< [grid::tick](#), [def::coordf](#) > [point1df](#)

Type alias for a point defined on the 1D ruler with floating-point coordinates.

- `typedef point< mln::grid::square, mln::def::coordf > point2df`
Type alias for a point defined on the 2D square grid with floating-point coordinates.
- `typedef point< grid::cube, def::coordf > point3df`
Type alias for a point defined on the 3D square grid with floating-point coordinates.
- `typedef mln::complex_image< 2, mln::space_2complex_geometry, mln::value::rgb8 > rgb8_2complex_image3df`
Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- `typedef mln::geom::complex_geometry< 2, point3df > space_2complex_geometry`
Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).
- `typedef mln::complex_image< 2, mln::space_2complex_geometry, unsigned > unsigned_2complex_image3df`
Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- `typedef algebra::vec< 2u, double > vec2d_d`
2D vector with double coordinates.
- `typedef algebra::vec< 2u, float > vec2d_f`
2D vector with float coordinates.
- `typedef algebra::vec< 3u, double > vec3d_d`
3D vector with double coordinates.
- `typedef algebra::vec< 3u, float > vec3d_f`
3D vector with float coordinates.
- `typedef window< mln::dpoint1d > window1d`
Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.
- `typedef window< mln::dpoint2d > window2d`
Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.
- `typedef window< mln::dpoint3d > window3d`
Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.
- `typedef point< grid::tick, def::coord > point1d`
Type alias for a point defined on the 1D ruler with integer coordinates.
- `typedef point< mln::grid::square, mln::def::coord > point2d`
Type alias for a point defined on the 2D square grid with integer coordinates.
- `typedef point< grid::hexa, def::coord > point2d_h`

Type alias for a point defined on the 2D hexagonal grid with integer coordinates.

- typedef [point](#)< [grid::cube](#), [def::coord](#) > [point3d](#)

Type alias for a point defined on the 3D square grid with integer coordinates.

Functions

- template<typename I >
I::psite [a_point_of](#) (const [Image](#)< I > &ima)
Give a point of an image.
- template<typename I , typename F >
[p2p_image](#)< I, F > [apply_p2p](#) ([Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- template<typename I , typename F >
[p2p_image](#)< const I, F > [apply_p2p](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- const [neighb3d](#) & [c18](#) ()
18-connectivity neighborhood on the 3D grid.
- const [neighb1d](#) & [c2](#) ()
2-connectivity neighborhood on the 1D grid.
- const [neighb3d](#) & [c26](#) ()
26-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c2_3d_sli](#) ()
depth 2-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c2_col](#) ()
Vertical 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c2_row](#) ()
Horizontal 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c4](#) ()
4-connectivity neighborhood on the 2D grid.
- const [neighb3d](#) & [c4_3d](#) ()
4-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c6](#) ()
6-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c8](#) ()
8-connectivity neighborhood on the 2D grid.

- const [neighb3d](#) & [c8_3d](#) ()
8-connectivity neighborhood on the 3D grid.
- template<typename T2 , typename T1 >
[fun::x2x::composed](#)< T2, T1 > [compose](#) (T2 f, T1 g)
Do a composition of two transformations.
- template<typename I >
[mln::trait::concrete](#)< I >::ret [duplicate](#) (const [Image](#)< I > &model)
Duplicate the image model with the values of the image data.
- template<typename I , typename F >
[extension_fun](#)< const I, F > [extend](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &fun)
Routines for domain extension with a function.
- template<typename I , typename J >
[extension_ima](#)< const I, const J > [extend](#) (const [Image](#)< I > &ima, const [Image](#)< J > &ext)
Routines for domain extension with an image.
- template<typename I >
[extension_val](#)< const I > [extend](#) (const [Image](#)< I > &ima, const typename I::value &val)
Routines for domain extension with a value.
- bool [implies](#) (bool lexpr, bool rexpr)
Implication.
- template<typename I , typename J >
void [initialize](#) ([Image](#)< I > &target, const [Image](#)< J > &model)
- template<typename P >
[box](#)< P > [larger_than](#) (const [box](#)< P > a, const [box](#)< P > b)
Return the minimum box including box a and box b.
- template<typename I , typename V , typename E >
[image2d](#)< typename I::value > [make_debug_graph_image](#) (const I &input, const V &ima_v, const E &ima_e, const [value::rgb8](#) &bg)
Draw a graph.
- [mln_gen_complex_neighborhood](#) (complex_higher_dim_connected_n_face_neighborhood, complex_higher_dim_connected_n_face_window)
Neighborhood centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.
- [mln_gen_complex_neighborhood](#) (complex_m_face_neighborhood, complex_m_face_window)
Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.
- [mln_gen_complex_neighborhood](#) (complex_lower_neighborhood, complex_lower_window)
Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces.
- [mln_gen_complex_neighborhood](#) (complex_higher_neighborhood, complex_higher_window)
Neighborhood centered on an n-face of complex returning its adjacent (n+1)-faces.

- [mln_gen_complex_neighborhood](#) (complex_lower_higher_neighborhood, complex_lower_higher_window)
Neighborhood centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces.
- [mln_gen_complex_neighborhood](#) (complex_lower_dim_connected_n_face_neighborhood, complex_lower_dim_connected_n_face_window)
Neighborhood centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_lower_window, [topo::adj_lower_face_fwd_iter](#), [topo::adj_lower_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces.
- [mln_gen_complex_window](#) (complex_higher_window, [topo::adj_higher_face_fwd_iter](#), [topo::adj_higher_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n+1)$ -faces.
- [mln_gen_complex_window](#) (complex_lower_higher_window, [topo::adj_lower_higher_face_fwd_iter](#), [topo::adj_lower_higher_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces.
- [mln_gen_complex_window](#) (complex_lower_dim_connected_n_face_window, [topo::adj_lower_dim_connected_n_face_fwd_iter](#), [topo::adj_lower_dim_connected_n_face_bkd_iter](#))
Window centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_higher_dim_connected_n_face_window, [topo::adj_higher_dim_connected_n_face_fwd_iter](#), [topo::adj_higher_dim_connected_n_face_bkd_iter](#))
Window centered on an n -face of complex returning the n -faces sharing an $(n+1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_m_face_window, [topo::adj_m_face_fwd_iter](#), [topo::adj_m_face_bkd_iter](#))
Window centered on an n -face of complex returning the m -faces transitively adjacent to this center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_window_p, [topo::adj_lower_face_fwd_iter](#), [topo::adj_lower_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_higher_window_p, [topo::adj_higher_face_fwd_iter](#), [topo::adj_higher_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n+1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_higher_window_p, [topo::adj_lower_higher_face_fwd_iter](#), [topo::adj_lower_higher_face_bkd_iter](#))
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_dim_connected_n_face_window_p, [topo::adj_lower_dim_connected_n_face_fwd_iter](#), [topo::adj_lower_dim_connected_n_face_bkd_iter](#))
Window centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face, as well as this center n -face.

- `mln_gen_complex_window_p` (`complex_higher_dim_connected_n_face_window_p`, `topo::adj_higher_dim_connected_n_face_fwd_iter`, `topo::adj_higher_dim_connected_n_face_bkd_iter`)
Window centered on an n -face of complex returning the n -faces sharing an $(n+1)$ -face with the center n -face, as well as this center n -face.
- `mln_gen_complex_window_p` (`complex_m_face_window_p`, `topo::adj_m_face_fwd_iter`, `topo::adj_m_face_bkd_iter`)
Window centered on an n -face of complex returning the m -faces transitively adjacent to this center n -face, as well as this center n -face.
- `template<typename W1 , typename W2 >`
`mln_regular` (`W1`) `operator-(const Window< W1 > &win1`
Set difference between a couple of windows `win1` and `win2`.
- `template<typename O1 , typename O2 >`
`mln_trait_op_geq` (`O1`, `O2`) `operator>`
General definition of the "greater than or equal to" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_greater` (`O1`, `O2`) `operator>`(`const Object< O1 > &lhs`
General definition of the "greater than" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_leq` (`O1`, `O2`) `operator<`
Default definition of the "less than or equal to" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_neq` (`O1`, `O2`) `operator!`
General definition of the "not equal to" operator.
- `template<typename P , typename S >`
`P operator*` (`const Gpoint< P > &p`, `const value::scalar_< S > &s`)
Multiply a point `p` by a scalar `s`.
- `template<typename S >`
`S & operator++` (`value::Scalar< S > &rhs`)
Pre-incrementation for any scalar type.
- `template<typename P , typename D >`
`P operator-` (`const Gpoint< P > &p`, `const Gdpoint< D > &dp`)
Subtract a delta-point `dp` to a grid point `p`.
- `template<typename N1 , typename N2 >`
`neighb< typename N1::window::regular > operator-` (`const Neighborhood< N1 > &nbh1`, `const Neighborhood< N2 > &nbh2`)
Set difference between a couple of neighborhoods `nbh1` and `nbh2`.
- `template<typename S >`
`S & operator--` (`value::Scalar< S > &rhs`)
Pre-decrementation for any scalar type.

- `template<typename L , typename R >`
`bool operator< (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise test if the pixel values of lhs are point-wise less than the pixel values of rhs.
- `template<typename I , typename G , typename W >`
`std::ostream & operator<< (std::ostream &ostr, const complex_window_bkd_piter< I, G, W > &p)`
Print an `mln::complex_window_bkd_piter`.
- `template<typename I , typename G , typename W >`
`std::ostream & operator<< (std::ostream &ostr, const complex_window_fwd_piter< I, G, W > &p)`
Print an `mln::complex_window_fwd_piter`.
- `template<typename I , typename G , typename N >`
`std::ostream & operator<< (std::ostream &ostr, const complex_neighborhood_bkd_piter< I, G, N > &p)`
Print an `mln::complex_neighborhood_bkd_piter`.
- `template<typename I , typename G , typename N >`
`std::ostream & operator<< (std::ostream &ostr, const complex_neighborhood_fwd_piter< I, G, N > &p)`
Print an `mln::complex_neighborhood_fwd_piter`.
- `template<unsigned D, typename G >`
`bool operator<= (const p_complex< D, G > &lhs, const p_complex< D, G > &rhs)`
Inclusion of a `mln::p_complex` in another one.
- `template<typename G , typename F >`
`bool operator<= (const p_vertices< G, F > &lhs, const p_vertices< G, F > &rhs)`
Inclusion of a `mln::p_vertices` in another one.
- `template<typename L , typename R >`
`bool operator<= (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise test if the pixel values of lhs are point-wise less than or equal to the pixel values of rhs.
- `template<typename G , typename F >`
`bool operator<= (const p_edges< G, F > &lhs, const p_edges< G, F > &rhs)`
Inclusion of a `mln::p_edges` in another one.
- `template<unsigned N, unsigned D, typename P >`
`bool operator<= (const p_faces< N, D, P > &lhs, const p_faces< N, D, P > &rhs)`
Inclusion of a `mln::p_faces` in another one.
- `template<typename G , typename F >`
`bool operator== (const p_edges< G, F > &lhs, const p_edges< G, F > &rhs)`
Comparison between two `mln::p_edges`'s.
- `template<typename L , typename R >`
`bool operator== (const Image< L > &lhs, const Image< R > &rhs)`

Point-wise test if the pixel values of lhs are equal to the pixel values of rhs.

- template<unsigned D, typename G >
bool [operator==](#) (const [p_complex](#)< D, G > &lhs, const [p_complex](#)< D, G > &rhs)
Comparison between two [mln::p_complex](#)'s.
- template<unsigned N, unsigned D, typename P >
bool [operator==](#) (const [p_faces](#)< N, D, P > &lhs, const [p_faces](#)< N, D, P > &rhs)
Comparison between two [mln::p_faces](#)'s.
- template<typename G, typename F >
bool [operator==](#) (const [p_vertices](#)< G, F > &lhs, const [p_vertices](#)< G, F > &rhs)
Comparison between two [mln::p_vertices](#)'s.
- template<typename S, typename F >
[p_if](#)< S, F > [operator|](#) (const [Site_Set](#)< S > &s, const [Function_v2b](#)< F > &f)
Restrict a site set s to points that verify f.
- template<typename V, typename G, typename P >
[edge_image](#)< P, V, G > [operator|](#) (const fun::i2v::array< V > &edge_values, const [p_edges](#)< G, fun::i2v::array< P > > &pe)
Construct a edge image from a fun::i2v::array and a [p_edges](#).
- template<typename I, typename F >
[image_if](#)< I, F > [operator|](#) ([Image](#)< I > &ima, const [Function_v2b](#)< F > &f)
ima | f creates an image_if with the image ima and the function f.
- template<typename V, typename G, typename P >
[vertex_image](#)< P, V, G > [operator|](#) (const fun::i2v::array< V > &vertex_values, const [p_vertices](#)< G, fun::i2v::array< P > > &pv)
Construct a vertex image from a fun::i2v::array and a [p_vertices](#).
- template<typename I, typename F >
[image_if](#)< const I, F > [operator|](#) (const [Image](#)< I > &ima, const [Function_v2b](#)< F > &f)
ima | f creates an image_if with the image ima and the function f.
- template<typename F, typename S >
[pw::image](#)< F, S > [operator|](#) (const [Function_v2v](#)< F > &f, const [Site_Set](#)< S > &ps)
Construct an image from a function and a site set.
- template<typename I >
const internal::primary_type< I >::ret & [primary](#) (const [Image](#)< I > &input)
FIXME: Doc!
- template<typename S, typename F >
[p_transformed](#)< S, F > [ptransform](#) (const [Site_Set](#)< S > &s, const [Function_v2v](#)< F > &f)
Transform a site set s through the function f.
- const [window2d](#) & [win_c4p](#) ()
4-connectivity window on the 2D grid, including the center.

- const [window3d](#) & [win_c4p_3d](#) ()
4-connectivity window on the 3D grid, including the center.
- const [window2d](#) & [win_c8p](#) ()
8-connectivity window on the 2D grid, including the center.
- const [window3d](#) & [win_c8p_3d](#) ()
8-connectivity window on the 3D grid, including the center.
- template<unsigned N, unsigned D, typename P >
bool [operator==](#) (const faces_psite< N, D, P > &lhs, const faces_psite< N, D, P > &rhs)
Comparison of two instances of mln::faces_psite.
- template<unsigned N, unsigned D, typename P >
bool [operator!=](#) (const faces_psite< N, D, P > &lhs, const faces_psite< N, D, P > &rhs)
Is lhs equal to rhs?
- template<unsigned N, unsigned D, typename P >
bool [operator<](#) (const faces_psite< N, D, P > &lhs, const faces_psite< N, D, P > &rhs)
Is lhs "less" than rhs?
- template<typename T >
[mln_exact](#) (T)*exact(T *ptr)
Exact cast routine for mln objects.
- template<unsigned D, typename G >
bool [operator==](#) (const [complex_psite](#)< D, G > &lhs, const [complex_psite](#)< D, G > &rhs)
Comparison of two instances of mln::complex_psite.
- template<unsigned D, typename G >
bool [operator!=](#) (const [complex_psite](#)< D, G > &lhs, const [complex_psite](#)< D, G > &rhs)
Is lhs not equal to rhs?
- template<unsigned D, typename G >
bool [operator<](#) (const [complex_psite](#)< D, G > &lhs, const [complex_psite](#)< D, G > &rhs)
Is lhs "less" than rhs?

Variables

- const [dpoint1d before](#) = [dpoint1d](#)(-1)
Definition of a shortcut for delta point in 1d.
- const [dpoint2d up](#) = [dpoint2d](#)(-1, 0)
Definition of a shortcut for delta point in 2d.
- const [dpoint3d sagittal_dec](#) = [dpoint3d](#)(0, 0, -1)
Definition of a shortcut for delta point in 3d.

9.1.1 Detailed Description

[mln/convert/to_image.hh](#) This implementation is not an usual heap, it allows to set an error rate so that some nodes may be "corrupted".

Generic class for hierarchical queues.

Merge with [mln/topo/skeleton/is_simple_point.hh](#) The fastest version give an approximation of the result.

The generic dual input tree algorithm for high quantized image.

The dual input tree algorithm specialized for low quantized image.

[mln/linear/convolve_directional.hh](#)

Read AVS header from a file.

Define a function which aborts a process in io module.

Forward declaration.

[mln/core/def/all.hh](#)

The namespace mln corresponds to the Milena (mini-Olena) project.

This accumulator uses an [mln::util::pix](#) (pixel) to update the reference level, area and volume information of the component.

The class mln/accu/volume is not a general-purpose accumulator; it is used to implement volume-based connected filters.

See also

[mln::morpho::closing::volume](#)
[mln::morpho::opening::volume](#)

The functor should provide the following methods:

- `template <typename g>=""> void init(const Graph<G>& g)` Will be called at the beginning.
- `bool to_be_treated(unsigned id)` Return whether this vertex has already been marked or if it may be a component representative.
- `void new_component_from_vertex(unsigned id)` will be called for the first vertex encountered for each component.
- `void process_vertex(unsigned id)` Will be called for each vertex queued.
- `bool to_be_queued(unsigned id)` Return whether this vertex has already been marked or if it can be added to the current component.
- `void added_to_queue(unsigned id)` Will be called for every vertex encountered in each component, except the first one.
- `void next_component()` Will be called after all vertices from a component have been treated.
- `void final()` Will be called at the end;

Conversions to [mln::Image](#).

FIXME: Re-write this description.

The contents of mln mimics the contents of the olena project but in a simplified way. Some classes have the same name in both projects and roughly have the same behavior.

Warning

The Milena project is independent from the Olena project; the user has to choose between both the project she wants to work with.

File that includes all core definitions.

The set of operators defined in this file is:

```

l += r : l = l + r, -> l&
l -= r : l = l - r, -> l&
l *= r : l = l * r, -> l&
l /= r : l = l / r, -> l&
l %= r : l = l % r, -> l&

+ r    : -> r
- r    : -> (0 - r)

l ++   : t = l, ++l, -> t
l --   : t = l, --l, -> t

++ r   : r += 1, -> r&
-- r   : r -= 1, -> r&

l != r : -> ! (l == r)

l > r  : -> (r < l)
l >= r : -> (r <= l)
l <= r : -> ! (r < l)    warning: re-define when partial ordering

```

As a consequence, the set of operators to be defined along with a client class is:

```

l + r
l - r
l * r
l / r

l == r

l < r
l <= r in case of partial ordering

```

Convolution by a line-shaped (directional) kernel.

This implementation is based on P. Salembier algorithm using hierarchical queues. This implies a low-quantized input image so that the number of queues is limited.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

TODO: During the canonization pass, we build the tree site set from the sorted site set of f, so that we compute twice f histogram (can be avoided).

This implementation is based on tarjan's union method, so that image quantization does not impact on the computation time.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

Do we want it to be exact?

Hierarchical queues are often used with connected operators (P. Salembier's max tree algorithm relies on these queues). To be efficient, the hierarchy is a static array and each are preallocated using an histogram.

FIXME: consider hqueues as a site set ?

A "corrupted node" means that its correct order is not totally preserved for performance reasons. Of course, it will have an impact on the returned values. As a result, be ware of not using this data structure if the element order is relevant for to you.

A corruption threshold can be passed to the constructor. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced. Tuning this threshold may have an impact on the structure entropy thus on the returned values order. It may also have an impact on the performance.

More implementation details are available in: "The soft heap: an approximate priority queue with optimal error rate", Bernard Chazelle, JACM, 2000.

URL: <http://www.cs.princeton.edu/~chazelle/pubs/sheap.pdf>

9.1.2 Typedef Documentation

9.1.2.1 `typedef mln::complex_image<1, mln::discrete_plane_1complex_geometry, bool> mln::bin_1complex_image2d`

Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 53 of file mln/core/alias/complex_image.hh.

9.1.2.2 `typedef mln::complex_image<2, mln::space_2complex_geometry, bool> mln::bin_2complex_image3df`

Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 85 of file mln/core/alias/complex_image.hh.

9.1.2.3 `typedef box<mln::point1d> mln::box1d`

Type alias for a box defined on the 1D square grid with integer coordinates.

See also

`mln::win::rectangle1d`.

Definition at line 47 of file core/alias/box1d.hh.

9.1.2.4 `typedef box<mln::point2d> mln::box2d`

Type alias for a box defined on the 2D square grid with integer coordinates.

See also

[mln::win::rectangle2d](#).

Definition at line 45 of file core/alias/box2d.hh.

9.1.2.5 typedef box<point2d_h> mln::box2d_h

FIXME.

Definition at line 47 of file core/alias/box2d_h.hh.

9.1.2.6 typedef box<point3d> mln::box3d

Type alias for a box defined on the 3D square grid with integer coordinates.

See also

[mln::win::rectangle3d](#).

Definition at line 45 of file core/alias/box3d.hh.

9.1.2.7 typedef mln::geom::complex_geometry<1, point2d> mln::discrete_plane_1complex_geometry

Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 45 of file core/alias/complex_geometry.hh.

9.1.2.8 typedef mln::geom::complex_geometry<2, point2d> mln::discrete_plane_2complex_geometry

Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 50 of file core/alias/complex_geometry.hh.

9.1.2.9 typedef dpoint<mln::grid::tick, def::coord> mln::dpoint1d

Type alias for a delta-point defined on the 1D square grid with integer coordinates.

Definition at line 44 of file dpoint1d.hh.

9.1.2.10 typedef dpoint<mln::grid::square, mln::def::coord> mln::dpoint2d

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 44 of file dpoint2d.hh.

9.1.2.11 typedef dpoint<mln::grid::hexa, def::coord> mln::dpoint2d_h

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file core/alias/dpoint2d_h.hh.

9.1.2.12 typedef dpoint<mln::grid::cube, def::coord> mln::dpoint3d

Type alias for a delta-point defined on the 3D square grid with integer coordinates.

Definition at line 43 of file dpoint3d.hh.

9.1.2.13 typedef mln::complex_image<2, mln::space_2complex_geometry, float> mln::float_2complex_image3df

Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 106 of file mln/core/alias/complex_image.hh.

9.1.2.14 typedef mln::complex_image<1, mln::discrete_plane_1complex_geometry, mln::value::int_u8> mln::int_u8_1complex_image2d

Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 61 of file mln/core/alias/complex_image.hh.

9.1.2.15 typedef mln::complex_image<2, mln::discrete_plane_2complex_geometry, mln::value::int_u8> mln::int_u8_2complex_image2d

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 74 of file mln/core/alias/complex_image.hh.

9.1.2.16 typedef mln::complex_image<2, mln::space_2complex_geometry, mln::value::int_u8> mln::int_u8_2complex_image3df

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 92 of file mln/core/alias/complex_image.hh.

9.1.2.17 typedef p_run<point2d> mln::p_run2d

Type alias for a run of 2d points.

Definition at line 42 of file p_run2d.hh.

9.1.2.18 typedef p_set_of<p_run2d> mln::p_runs2d

Type alias for a set of runs of 2d points.

Definition at line 42 of file p_runs2d.hh.

9.1.2.19 typedef point< grid::tick, def::coord > mln::point1d

Type alias for a point defined on the 1D ruler with integer coordinates.

Definition at line 45 of file point1d.hh.

9.1.2.20 typedef point<grid::tick, def::coordf> mln::point1df

Type alias for a point defined on the 1D ruler with floating-point coordinates.

Definition at line 49 of file point1d.hh.

9.1.2.21 typedef point< grid::square, def::coord > mln::point2d

Type alias for a point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file point2d.hh.

9.1.2.22 typedef point< grid::hexa, def::coord > mln::point2d_h

Type alias for a point defined on the 2D hexagonal grid with integer coordinates.

Definition at line 43 of file core/alias/point2d_h.hh.

9.1.2.23 typedef point<mln::grid::square, mln::def::coordf> mln::point2df

Type alias for a point defined on the 2D square grid with floating-point coordinates.

Definition at line 49 of file point2d.hh.

9.1.2.24 typedef point< grid::cube, def::coord > mln::point3d

Type alias for a point defined on the 3D square grid with integer coordinates.

Definition at line 44 of file point3d.hh.

9.1.2.25 typedef point<grid::cube, def::coordf> mln::point3df

Type alias for a point defined on the 3D square grid with floating-point coordinates.

Definition at line 48 of file point3d.hh.

9.1.2.26 `typedef mln::complex_image<2, mln::space_2complex_geometry, mln::value::rgb8>
mln::rgb8_2complex_image3df`

Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 113 of file mln/core/alias/complex_image.hh.

9.1.2.27 `typedef mln::geom::complex_geometry<2, point3df> mln::space_2complex_geometry`

Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).

Definition at line 54 of file core/alias/complex_geometry.hh.

9.1.2.28 `typedef mln::complex_image<2, mln::space_2complex_geometry, unsigned>
mln::unsigned_2complex_image3df`

Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 99 of file mln/core/alias/complex_image.hh.

9.1.2.29 `typedef algebra::vec<2u,double> mln::vec2d_d`

2D vector with double coordinates.

Definition at line 43 of file vec2d.hh.

9.1.2.30 `typedef algebra::vec<2u,float> mln::vec2d_f`

2D vector with float coordinates.

Definition at line 40 of file vec2d.hh.

9.1.2.31 `typedef algebra::vec<3u,double> mln::vec3d_d`

3D vector with double coordinates.

Definition at line 43 of file vec3d.hh.

9.1.2.32 `typedef algebra::vec<3u,float> mln::vec3d_f`

3D vector with float coordinates.

Definition at line 40 of file vec3d.hh.

9.1.3 Function Documentation

9.1.3.1 `template<typename I> I::psite mln::a_point_of(const Image< I> & ima) [inline]`

Give a point of an image.

Definition at line 51 of file a_point_of.hh.

9.1.3.2 `template<typename I , typename F > p2p_image< I, F > mln::apply_p2p (Image< I > & ima, const Function_v2v< F > & f) [inline]`

FIXME: Doc!

Definition at line 242 of file p2p_image.hh.

Referenced by mln::debug::mosaic(), and mln::debug::slices_2d().

9.1.3.3 `template<typename I , typename F > p2p_image< const I, F > mln::apply_p2p (const Image< I > & ima, const Function_v2v< F > & f) [inline]`

FIXME: Doc!

Definition at line 256 of file p2p_image.hh.

9.1.3.4 `template<typename T2 , typename T1 > fun::x2x::composed< T2, T1 > mln::compose (T2 f, T1 g) [inline]`

Do a composition of two transformations.

Parameters

[in] *f* The second transformation.

[in] *g* The first transformation.

Returns

The composed transformation fog.

Definition at line 306 of file composed.hh.

References compose().

Referenced by compose(), and mln::geom::rotate().

9.1.3.5 `template<typename I > mln::trait::concrete< I >::ret mln::duplicate (const Image< I > & model) [inline]`

Duplicate the image `model` with the values of the image `data`.

Parameters

[in] *model* The image to be duplicated.

Returns

The duplicate.

Precondition

`model.is_valid`

Definition at line 56 of file core/routine/duplicate.hh.

References `mln::data::fill()`, and `initialize()`.

Referenced by `mln::registration::icp()`, `mln::plain< I >::operator I()`, `mln::geom::rotate()`, `mln::geom::impl::seeds2tiling()`, and `mln::labeling::superpose()`.

9.1.3.6 `template<typename I, typename F> extension_fun< const I, F> mln::extend (const Image< I> & ima, const Function_v2v< F> & fun) [inline]`

Routines for domain extension with a function.

Definition at line 140 of file extend.hh.

Referenced by `mln::geom::translate()`.

9.1.3.7 `template<typename I, typename J> extension_ima< const I, const J> mln::extend (const Image< I> & ima, const Image< J> & ext)`

Routines for domain extension with an image.

Definition at line 162 of file extend.hh.

9.1.3.8 `template<typename I> extension_val< const I> mln::extend (const Image< I> & ima, const typename I::value & val) [inline]`

Routines for domain extension with a value.

Definition at line 175 of file extend.hh.

9.1.3.9 `bool mln::implies (bool lexpr, bool rexpr) [inline]`

Implication.

Definition at line 68 of file contract.hh.

Referenced by `mln::p_line2d::is_valid()`.

9.1.3.10 `template<typename I, typename J> void mln::initialize (Image< I> & target, const Image< J> & model) [inline]`

Initialize the image `target` with data extracted from image `model`.

Parameters

- [in, out] ***target*** The image to be initialized.
- [in] ***model*** The image to provide data for the initialization.

Precondition

(not `target.is_valid()` and `model.is_valid()`)

Definition at line 56 of file initialize.hh.

Referenced by `duplicate()`, `mln::histo::equalize()`, `mln::morpho::tree::filter::filter()`, `mln::linear::gaussian()`, `mln::linear::gaussian_1st_derivative()`, `mln::linear::gaussian_2nd_derivative()`,

mln::graph::labeling(), mln::io::magick::load(), mln::io::dicom::load(), make_debug_graph_image(), mln::morpho::tree::filter::max(), mln::morpho::meyer_wst(), mln::morpho::tree::filter::min(), mln::arith::min(), mln::arith::minus(), mln::arith::plus(), mln::arith::revert(), mln::geom::rotate(), mln::data::impl::stretch(), mln::morpho::watershed::topological(), and mln::data::impl::generic::transform().

9.1.3.11 `template<typename P> box< P> mln::larger_than (const box< P> a, const box< P> b) [inline]`

Return the minimum box including box a and box b.

Definition at line 362 of file core/site_set/box.hh.

References mln::box< P>::pmax(), and mln::box< P>::pmin().

9.1.3.12 `template<typename I, typename V, typename E> image2d<typename I::value> mln::make_debug_graph_image (const I & input, const V & ima_v, const E & ima_e, const value::rgb8 & bg) [inline]`

Draw a graph.

Definition at line 123 of file demos/graph/region_adjacency_graph.cc.

References mln::box< P>::crop_wrt(), mln::image2d< T>::domain(), mln::debug::draw_graph(), mln::data::fill(), mln::literal::green, initialize(), and mln::convert::to().

9.1.3.13 `template<typename T> mln::mln_exact (T) [inline]`

Exact cast routine for mln objects.

This set of routines can be used to downcast an object towards its exact type. The only argument, respectively `ptr` or `ref`, should be an [mln::Object](#).

The parameter E is the exact type of the object.

Returns

The return follows the nature of the argument (either a pointer or a reference, const or not).

Definition at line 92 of file routine/exact.hh.

Referenced by mln::geom::rotate(), mln::Accumulator< E>::take_as_init(), mln::Accumulator< E>::take_n_times(), mln::convert::to(), and mln::geom::translate().

9.1.3.14 `mln::mln_gen_complex_neighborhood (complex_higher_neighborhood, complex_higher_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

9.1.3.15 `mln::mln_gen_complex_neighborhood (complex_lower_higher_neighborhood, complex_lower_higher_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

9.1.3.16 `mln::mln_gen_complex_neighborhood (complex_lower_dim_connected_n_face_neighborhood , complex_lower_dim_connected_n_face_window)`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

9.1.3.17 `mln::mln_gen_complex_neighborhood (complex_higher_dim_connected_n_face_neighborhood , complex_higher_dim_connected_n_face_window)`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

9.1.3.18 `mln::mln_gen_complex_neighborhood (complex_m_face_neighborhood , complex_m_face_window)`

[Neighborhood](#) centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

9.1.3.19 `mln::mln_gen_complex_neighborhood (complex_lower_neighborhood , complex_lower_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

9.1.3.20 `mln::mln_gen_complex_window (complex_lower_window , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

9.1.3.21 `mln::mln_gen_complex_window (complex_higher_window , topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

9.1.3.22 `mln::mln_gen_complex_window (complex_lower_higher_window , topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

9.1.3.23 `mln::mln_gen_complex_window (complex_lower_dim_connected_n_face_window , topo::adj_lower_dim_connected_n_face_fwd_iter , topo::adj_lower_dim_connected_n_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

9.1.3.24 `mln::mln_gen_complex_window (complex_higher_dim_connected_n_face_window , topo::adj_higher_dim_connected_n_face_fwd_iter , topo::adj_higher_dim_connected_n_face_bkd_iter)`

Window centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

9.1.3.25 `mln::mln_gen_complex_window (complex_m_face_window , topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter)`

Window centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

9.1.3.26 `mln::mln_gen_complex_window_p (complex_lower_window_p , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter)`

Window centered on an n-face of complex returning its adjacent (n-1)-faces as well as the center n-face.

9.1.3.27 `mln::mln_gen_complex_window_p (complex_higher_window_p , topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter)`

Window centered on an n-face of complex returning its adjacent (n+1)-faces as well as the center n-face.

9.1.3.28 `mln::mln_gen_complex_window_p (complex_lower_higher_window_p , topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter)`

Window centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces as well as the center n-face.

9.1.3.29 `mln::mln_gen_complex_window_p (complex_higher_dim_connected_n_face_window_p , topo::adj_higher_dim_connected_n_face_fwd_iter , topo::adj_higher_dim_connected_n_face_bkd_iter)`

Window centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face, as well as this center n-face.

9.1.3.30 `mln::mln_gen_complex_window_p (complex_lower_dim_connected_n_face_window_p , topo::adj_lower_dim_connected_n_face_fwd_iter , topo::adj_lower_dim_connected_n_face_bkd_iter)`

Window centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face, as well as this center n-face.

9.1.3.31 `mln::mln_gen_complex_window_p (complex_m_face_window_p , topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter)`

Window centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face, as well as this center n-face.

9.1.3.32 `template<typename W1 , typename W2 > mln::mln_regular (W1) const [inline]`

Set difference between a couple of windows `win1` and `win2`.

Inter a window `win` with a delta-point `dp`.

It just calls `mln::win::diff`.

9.1.3.33 `template<typename O1 , typename O2 > mln::mln_trait_op_geq (O1 , O2)`

General definition of the "greater than or equal to" operator.

The "greater than or equal to" operator is here defined for every Milena objects. It relies on the definition of the "less than or equal to" operator. It returns "`rhs <= lhs`".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of `mln::Object`.

9.1.3.34 `template<typename O1 , typename O2 > mln::mln_trait_op_greater (O1 , O2) const`

General definition of the "greater than" operator.

The "greater than" operator is here defined for every milena objects. It relies on the definition of the "less than" operator. It returns "`rhs < lhs`".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of `mln::Object`.

9.1.3.35 `template<typename O1 , typename O2 > mln::mln_trait_op_leq (O1 , O2)`

Default definition of the "less than or equal to" operator.

A default version of the "less than or equal to" operator is defined for every Milena objects. It relies on the definition of the "less than" operator. It returns "`not (rhs < lhs)`".

Warning

In the case of partial ordering between objects, this operator has to be re-defined.

**9.1.3.36 `template<typename O1 , typename O2 > mln::mln_trait_op_neq (O1 , O2)`
`[inline]`****Initial value:**

```
(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return ! (exact(lhs) == exact(rhs));
}
```

```

template <typename O1, typename O2>
inline
mln_trait_op_greater(O1, O2)
operator>(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return exact(rhs) < exact(lhs);
}

template <typename O1

```

General definition of the "not equal to" operator.

The "not equal to" operator is here defined for every milena objects. It relies on the definition of the "equal to" operator. It returns "not (lhs == rhs)".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of [mln::Object](#).

9.1.3.37 `template<unsigned D, typename G > bool mln::operator!=(const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs)`

Is *lhs* not equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

Definition at line 353 of file complex_psite.hh.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.38 `template<unsigned N, unsigned D, typename P > bool mln::operator!=(const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 336 of file faces_psite.hh.

9.1.3.39 `template<typename P , typename S > P mln::operator*(const Gpoint< P > & p, const value::scalar_< S > & s) [inline]`

Multiply a point *p* by a scalar *s*.

Definition at line 405 of file gpoint.hh.

9.1.3.40 `template<typename S > S & mln::operator++ (value::Scalar< S > & rhs)`
[inline]

Pre-incrementation for any scalar type.

Definition at line 77 of file concept/scalar.hh.

References `mln::literal::one`.

9.1.3.41 `template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular >`
`mln::operator- (const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > &`
`nbh2)`

Set difference between a couple of neighborhoods `nbh1` and `nbh2`.

It just calls `mln::win::diff`.

Definition at line 155 of file win/diff.hh.

References `mln::win::diff()`.

9.1.3.42 `template<typename P , typename D > P mln::operator- (const Gpoint< P > & p, const`
`Gdpoint< D > & dp) [inline]`

Subtract a delta-point `dp` to a grid point `p`.

Parameters

`[in]` *p* A grid point.

`[in]` *dp* A delta-point.

The type of `dp` has to compatible with the type of `p`.

Returns

A point (temporary object).

See also

[mln::Gdpoint](#)

[mln::Gdpoint](#)

Definition at line 395 of file gpoint.hh.

9.1.3.43 `template<typename S > S & mln::operator-- (value::Scalar< S > & rhs) [inline]`

Pre-decrementation for any scalar type.

Definition at line 85 of file concept/scalar.hh.

References `mln::literal::one`.

9.1.3.44 `template<unsigned D, typename G > bool mln::operator< (const complex_psite< D, G`
`> & lhs, const complex_psite< D, G > & rhs)`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

Definition at line 362 of file `complex_psite.hh`.

9.1.3.45 `template<unsigned N, unsigned D, typename P > bool mln::operator< (const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting psites.

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 345 of file `faces_psite.hh`.

9.1.3.46 `template<typename L , typename R > bool mln::operator< (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise test if the pixel values of *lhs* are point-wise less than the pixel values of *rhs*.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 107 of file `compare.hh`.

References `mln::test::predicate()`.

9.1.3.47 `template<typename I , typename G , typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_fwd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_fwd_piter](#).

Definition at line 292 of file `complex_window_piter.hh`.

9.1.3.48 `template<typename I , typename G , typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_bkd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_bkd_piter](#).

Definition at line 401 of file `complex_window_piter.hh`.

9.1.3.49 `template<typename I , typename G , typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_fwd_piter< I, G, N > & p)`
`[inline]`

Print an [mln::complex_neighborhood_fwd_piter](#).

Definition at line 292 of file `complex_neighborhood_piter.hh`.

9.1.3.50 `template<typename I , typename G , typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_bkd_piter< I, G, N > & p)`
`[inline]`

Print an [mln::complex_neighborhood_bkd_piter](#).

Definition at line 399 of file `complex_neighborhood_piter.hh`.

9.1.3.51 `template<typename G , typename F > bool mln::operator<= (const p_edges< G, F > & lhs, const p_edges< G, F > & rhs)`

Inclusion of a [mln::p_edges](#) in another one.

Definition at line 338 of file `p_edges.hh`.

9.1.3.52 `template<unsigned D, typename G > bool mln::operator<= (const p_complex< D, G > & lhs, const p_complex< D, G > & rhs)`

Inclusion of a [mln::p_complex](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p_complex](#) is included in another one if their are equal.

Definition at line 330 of file `p_complex.hh`.

9.1.3.53 `template<typename L , typename R > bool mln::operator<= (const Image< L > & lhs, const Image< R > & rhs)` `[inline]`

Point-wise test if the pixel values of `lhs` are point-wise less than or equal to the pixel values of `rhs`.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 126 of file `compare.hh`.

References `mln::test::predicate()`.

9.1.3.54 `template<unsigned N, unsigned D, typename P> bool mln::operator<= (const p_faces< N, D, P> & lhs, const p_faces< N, D, P> & rhs)`

Inclusion of a [mln::p_faces](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p_faces](#) is included in another one if their are equal.

Definition at line 287 of file p_faces.hh.

9.1.3.55 `template<typename G, typename F> bool mln::operator<= (const p_vertices< G, F> & lhs, const p_vertices< G, F> & rhs)`

Inclusion of a [mln::p_vertices](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for graphs is simple: a [mln::p_vertices](#) is included in another one if their are equal.

Definition at line 399 of file p_vertices.hh.

9.1.3.56 `template<unsigned N, unsigned D, typename P> bool mln::operator== (const p_faces< N, D, P> & lhs, const p_faces< N, D, P> & rhs)`

Comparison between two [mln::p_faces](#)'s.

Two [mln::p_faces](#)'s are considered equal if they share the same complex.

Definition at line 278 of file p_faces.hh.

References [mln::p_faces< N, D, P>::cplx\(\)](#).

9.1.3.57 `template<typename G, typename F> bool mln::operator== (const p_edges< G, F> & lhs, const p_edges< G, F> & rhs)`

Comparison between two [mln::p_edges](#)'s.

Two [mln::p_edges](#)'s are considered equal if they share the same graph.

Definition at line 331 of file p_edges.hh.

References [mln::p_edges< G, F>::graph\(\)](#).

9.1.3.58 `template<unsigned D, typename G> bool mln::operator== (const p_complex< D, G> & lhs, const p_complex< D, G> & rhs)`

Comparison between two [mln::p_complex](#)'s.

Two [mln::p_complex](#)'s are considered equal if they share the same complex.

Definition at line 321 of file p_complex.hh.

References [mln::p_complex< D, G>::cplx\(\)](#).

9.1.3.59 `template<unsigned D, typename G> bool mln::operator== (const complex_psite< D, G> & lhs, const complex_psite< D, G> & rhs)`

Comparison of two instances of [mln::complex_psite](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

Definition at line 344 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.60 `template<typename G , typename F > bool mln::operator==(const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs)`

Comparison between two [mln::p_vertices](#)'s.

Two [mln::p_vertices](#)'s are considered equal if they share the same graph.

Definition at line 392 of file `p_vertices.hh`.

References `mln::p_vertices< G, F >::graph()`.

9.1.3.61 `template<unsigned N, unsigned D, typename P > bool mln::operator==(const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Comparison of two instances of `mln::faces_psite`.

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 327 of file `faces_psite.hh`.

9.1.3.62 `template<typename L , typename R > bool mln::operator==(const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise test if the pixel values of *lhs* are equal to the pixel values of *rhs*.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 86 of file `compare.hh`.

References `mln::test::predicate()`.

9.1.3.63 `template<typename V , typename G , typename P > vertex_image<P,V,G> mln::operator|(const fun::i2v::array< V > & vertex_values, const p_vertices< G, fun::i2v::array< P > > & pv)`

Construct a vertex image from a `fun::i2v::array` and a [p_vertices](#).

image = fun::i2v::array | [p_vertices](#).

Definition at line 215 of file core/image/vertex_image.hh.

9.1.3.64 `template<typename V , typename G , typename P > edge_image<P,V,G> mln::operator| (const fun::i2v::array< V > & edge_values, const p_edges< G, fun::i2v::array< P > > & pe)`

Construct a edge image from a fun::i2v::array and a [p_edges](#).

image = fun::i2v::array | [p_edges](#).

Definition at line 217 of file core/image/edge_image.hh.

9.1.3.65 `template<typename I , typename F > image_if<const I,F> mln::operator| (const Image< I > & ima, const Function_v2b< F > & f)`

ima | f creates an image_if with the image ima and the function f.

Definition at line 242 of file image_if.hh.

9.1.3.66 `template<typename I , typename F > image_if<I,F> mln::operator| (Image< I > & ima, const Function_v2b< F > & f)`

ima | f creates an image_if with the image ima and the function f.

Definition at line 233 of file image_if.hh.

9.1.3.67 `template<typename F , typename S > pw::image<F,S> mln::operator| (const Function_v2v< F > & f, const Site_Set< S > & ps)`

Construct an image from a function and a site set.

image = function | site_set.

Definition at line 138 of file pw/image.hh.

9.1.3.68 `template<typename S , typename F > p_if<S, F> mln::operator| (const Site_Set< S > & s, const Function_v2b< F > & f)`

Restrict a site set s to points that verify f.

Parameters

[in] *s* A site set.

[in] *f* A function from point to Boolean.

Returns

A subset of points.

Definition at line 147 of file p_if.hh.

9.1.3.69 `template<typename I> const internal::primary_type< I >::ret & mln::primary (const Image< I> & input) [inline]`

FIXME: Doc!

Definition at line 129 of file primary.hh.

Referenced by `mln::border::resize()`.

9.1.3.70 `template<typename S , typename F> p_transformed< S, F> mln::pttransform (const Site_Set< S> & s, const Function_v2v< F> & f) [inline]`

Transform a site set *s* through the function *f*.

Parameters

[in] *s* A site set.

[in] *f* A function from site to site.

Returns

The transformed site set.

Definition at line 145 of file p_transformed.hh.

9.1.4 Variable Documentation

9.1.4.1 `const dpoint1d mln::before = dpoint1d(-1)`

Definition of a shortcut for delta point in 1d.

Definition at line 73 of file dpoint1d.hh.

9.1.4.2 `const dpoint3d mln::sagittal_dec = dpoint3d(0, 0, -1)`

Definition of a shortcut for delta point in 3d.

Definition at line 72 of file dpoint3d.hh.

9.1.4.3 `const dpoint2d mln::up = dpoint2d(-1, 0)`

Definition of a shortcut for delta point in 2d.

Definition at line 76 of file dpoint2d.hh.

9.2 mln::accu Namespace Reference

Namespace of accumulators.

Namespaces

- namespace [image](#)
Namespace of accumulator image routines.
- namespace [impl](#)
Implementation namespace of accumulator namespace.
- namespace [logic](#)
Namespace of logical accumulators.
- namespace [math](#)
Namespace of mathematic accumulators.
- namespace [shape](#)
Namespace of shape accumulators.
- namespace [stat](#)
Namespace of statistical accumulators.

Classes

- struct [center](#)
Mass center accumulator.
- struct [convolve](#)
Generic convolution accumulator class.
- struct [count_adjacent_vertices](#)
[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).
- struct [count_labels](#)
Count the number of different labels in an image.
- struct [count_value](#)
Define an accumulator that counts the occurrence of a given value.
- struct [histo](#)
Generic histogram class over a value set with type V .
- struct [label_used](#)
References all the labels used.
- struct [maj_h](#)
Compute the majority value.
- struct [max_site](#)
Define an accumulator that computes the first site with the maximum value in an image.

- struct [nil](#)
Define an accumulator that does nothing.
- struct [p](#)
Generic p of accumulators.
- struct [pair](#)
Generic pair of accumulators.
- struct [rms](#)
Generic root mean square accumulator class.
- struct [tuple](#)
Generic tuple of accumulators.
- struct [val](#)
Generic val of accumulators.

Functions

- template<typename A , typename I >
A::result [compute](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &input)
Make an accumulator compute the pixels of the image input.
- template<typename Meta_Accu , unsigned Dir, typename I , typename O >
void [line](#) (const [Image](#)< I > &input, const typename I::site &p_start, unsigned len, unsigned half_length, [Image](#)< O > &output)
- template<typename A , typename I >
[mln_meta_accu_result](#) (A, util::pix< I >) compute(const [Meta_Accumulator](#)< A > &a
Make an accumulator compute the pixels of the image input.
- template<typename A , typename I >
void [take](#) (const [Image](#)< I > &input, [Accumulator](#)< A > &a)
Make an accumulator take the pixels of the image input.

9.2.1 Detailed Description

Namespace of accumulators.

9.2.2 Function Documentation

- 9.2.2.1** template<typename A , typename I > A::result mln::accu::compute (const [Accumulator](#)< A > & a, const [Image](#)< I > & input) [inline]

Make an accumulator compute the pixels of the image input.

Parameters

- [in] *input* The input image.
- [in] *a* An accumulator.

This routine runs:

a.take(make::pix(input, p)); on all pixels on the images.

Warning

This routine does not perform a.init().

Definition at line 130 of file accu/compute.hh.

9.2.2.2 `template<typename Meta_Accu , unsigned Dir, typename I , typename O > void
mln::accu::line (const Image< I > & input, const typename I::site & p_start, unsigned
len, unsigned half_length, Image< O > & output)`

Line an accumulator onto the pixel values of the image *input*.

Parameters

- [in] *input* The input image.
- [in] *p_start* The starting site of the line.
- [in] *len* The line length.
- [in] *half_length* The half length of the line.
- [in, out] *output* The resulting image.

This routine runs:

```
tmp = a
tmp.init()
accu::take(input, tmp)
return tmp.to_result()
```

Definition at line 381 of file accu/line.hh.

9.2.2.3 `template<typename A , typename I > mln::accu::mln_meta_accu_result (A , util::pix< I
>) const [inline]`

Make an accumulator compute the pixels of the image *input*.

Parameters

- [in] *input* The input image.
- [in] *a* A meta accumulator.

This routine runs:

a.take(make::pix(input, p)); on all pixels on the images.

Warning

This routine does not perform a.init().

9.2.2.4 `template<typename A , typename I > void mln::accu::take (const Image< I > & input, Accumulator< A > & a) [inline]`

Make an accumulator take the pixels of the image `input`.

Parameters

[in] *input* The input image.

[in, out] *a* The accumulator.

This routine runs:

for all `p` of `input`, `a.take(pix(input, p))`

Warning

This routine does not perform `a.init()`.

Definition at line 87 of file `take.hh`.

9.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

9.3.1 Detailed Description

Namespace of accumulator image routines.

9.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

9.4.1 Detailed Description

Implementation namespace of accumulator namespace.

9.5 mln::accu::logic Namespace Reference

Namespace of logical accumulators.

Classes

- struct [land](#)
 "Logical-and" accumulator.
- struct [land_basic](#)

"Logical-and" accumulator.

- struct [lor](#)

"Logical-or" accumulator.

- struct [lor_basic](#)

"Logical-or" accumulator class.

9.5.1 Detailed Description

Namespace of logical accumulators.

9.6 mln::accu::math Namespace Reference

Namespace of mathematic accumulators.

Classes

- struct [count](#)

Generic counter accumulator.

- struct [inf](#)

Generic inf accumulator class.

- struct [sum](#)

Generic sum accumulator class.

- struct [sup](#)

Generic sup accumulator class.

9.6.1 Detailed Description

Namespace of mathematic accumulators.

9.7 mln::accu::meta::logic Namespace Reference

Namespace of logical meta-accumulators.

Classes

- struct [land](#)

Meta accumulator for land.

- struct [land_basic](#)

Meta accumulator for [land_basic](#).

- struct [lor](#)

Meta accumulator for [lor](#).

- struct [lor_basic](#)

Meta accumulator for [lor_basic](#).

9.7.1 Detailed Description

Namespace of logical meta-accumulators.

9.8 [mln::accu::meta::math](#) Namespace Reference

Namespace of mathematic meta-accumulators.

Classes

- struct [count](#)

Meta accumulator for [count](#).

- struct [inf](#)

Meta accumulator for [inf](#).

- struct [sum](#)

Meta accumulator for [sum](#).

- struct [sup](#)

Meta accumulator for [sup](#).

9.8.1 Detailed Description

Namespace of mathematic meta-accumulators.

9.9 [mln::accu::meta::shape](#) Namespace Reference

Namespace of shape meta-accumulators.

Classes

- struct [bbox](#)

Meta accumulator for [bbox](#).

- struct [height](#)

Meta accumulator for height.

- struct [volume](#)

Meta accumulator for volume.

9.9.1 Detailed Description

Namespace of shape meta-accumulators.

9.10 mln::accu::meta::stat Namespace Reference

Namespace of statistical meta-accumulators.

Classes

- struct [max](#)

Meta accumulator for max.

- struct [max_h](#)

Meta accumulator for max.

- struct [mean](#)

Meta accumulator for mean.

- struct [median_alt](#)

Meta accumulator for [median_alt](#).

- struct [median_h](#)

Meta accumulator for [median_h](#).

- struct [min](#)

Meta accumulator for min.

- struct [min_h](#)

Meta accumulator for min.

- struct [rank](#)

Meta accumulator for rank.

- struct [rank_high_quant](#)

Meta accumulator for [rank_high_quant](#).

9.10.1 Detailed Description

Namespace of statistical meta-accumulators.

9.11 mln::accu::shape Namespace Reference

Namespace of shape accumulators.

Classes

- struct [bbox](#)
Generic bounding box accumulator class.
- struct [height](#)
Height accumulator.
- struct [volume](#)
Volume accumulator class.

9.11.1 Detailed Description

Namespace of shape accumulators.

9.12 mln::accu::stat Namespace Reference

Namespace of statistical accumulators.

Classes

- struct [deviation](#)
Generic standard deviation accumulator class.
- struct [histo3d_rgb](#)
Define a histogram as accumulator which returns an [image3d](#).
- struct [max](#)
Generic max accumulator class.
- struct [max_h](#)
Generic max function based on histogram over a value set with type V .
- struct [mean](#)
Generic mean accumulator class.
- struct [median_alt](#)
Generic [median_alt](#) function based on histogram over a value set with type S .
- struct [median_h](#)
Generic median function based on histogram over a value set with type V .

- struct [min](#)
Generic min accumulator class.
- struct [min_h](#)
Generic min function based on histogram over a value set with type V.
- struct [min_max](#)
Generic min and max accumulator class.
- struct [rank](#)
Generic rank accumulator class.
- struct [rank< bool >](#)
rank accumulator class for Boolean.
- struct [rank_high_quant](#)
Generic rank accumulator class.
- struct [var](#)
Var accumulator class.
- struct [variance](#)
Variance accumulator class.

Functions

- `template<typename V > bool operator==(const histo3d_rgb< V > &histo1, const histo3d_rgb< V > &histo2)`
Check whether a histogram is equal to another one.

9.12.1 Detailed Description

Namespace of statistical accumulators.

9.12.2 Function Documentation

9.12.2.1 `template<typename V > bool mln::accu::stat::operator==(const histo3d_rgb< V > &histo1, const histo3d_rgb< V > &histo2)`

Check whether a histogram is equal to another one.

Parameters

- [in] *histo1* the first histogram to compare with.
- [in] *histo2* the second histogram.

The operator compares all the bins from the two histograms. Nobody uses this method unless unitary tests.

Definition at line 315 of file histo3d_rgb.hh.

References `mln::image3d< T >::domain()`.

9.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

Classes

- struct [h_mat](#)
N-Dimensional matrix with homogeneous coordinates.
- class [h_vec](#)
N-Dimensional vector with homogeneous coordinates.

Functions

- template<unsigned N, typename T >
bool [ldlt_decomp](#) (mat< N, N, T > &A, vec< N, T > &rdiag)
Perform LDL^T decomposition of a symmetric positive definite matrix.
- template<unsigned N, typename T >
void [ldlt_solve](#) (const mat< N, N, T > &A, const vec< N, T > &rdiag, const vec< N, T > &B, vec< N, T > &x)
Solve $Ax = B$ after [mln::algebra::ldlt_decomp](#).
- template<unsigned n, typename T, typename U >
mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum [operator*](#) (const vec< n, T > &lhs, const vec< n, U > &rhs)
Scalar product (dot product).
- template<typename T, typename U >
vec< 3, typename mln::trait::op::times< T, U >::ret > [vprod](#) (const vec< 3, T > &lhs, const vec< 3, U > &rhs)
Vectorial product (cross product).

9.13.1 Detailed Description

Namespace of algebraic structure.

9.13.2 Function Documentation

9.13.2.1 `template<unsigned N, typename T> bool mln::algebra::ldlt_decomp (mat< N, N, T > & A, vec< N, T > & rdiag) [inline]`

Perform LDL^T decomposition of a symmetric positive definite matrix.

Like Cholesky, but no square roots. Overwrites lower triangle of matrix.

From Trimesh's `ldltde` routine.

Definition at line 79 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

9.13.2.2 `template<unsigned N, typename T> void mln::algebra::ldlt_solve (const mat< N, N, T > & A, const vec< N, T > & rdiag, const vec< N, T > & B, vec< N, T > & x) [inline]`

Solve $Ax = B$ after [mln::algebra::ldlt_decomp](#).

Definition at line 112 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

9.13.2.3 `template<unsigned n, typename T, typename U> mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum mln::algebra::operator* (const vec< n, T > & lhs, const vec< n, U > & rhs) [inline]`

Scalar product (dot product).

Definition at line 632 of file `algebra/vec.hh`.

References `mln::literal::zero`.

9.13.2.4 `template<typename T, typename U> vec< 3, typename mln::trait::op::times< T, U >::ret > mln::algebra::vprod (const vec< 3, T > & lhs, const vec< 3, U > & rhs) [inline]`

Vectorial product (cross product).

Definition at line 713 of file `algebra/vec.hh`.

References `vprod()`.

Referenced by `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_curvature()`, `mln::geom::mesh_normal()`, and `vprod()`.

9.14 mln::arith Namespace Reference

Namespace of arithmetic.

Namespaces

- namespace [impl](#)

Implementation namespace of arith namespace.

Functions

- `template<typename I >`
`mln::trait::concrete< I >::ret diff_abs (const Image< I > &lhs, const Image< I > &rhs)`
Point-wise absolute difference of images lhs and rhs.
- `template<typename L , typename R , typename O >`
`void div (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`
Point-wise division of images lhs and rhs.
- `template<typename I , typename V , typename O >`
`void div_cst (const Image< I > &input, const V &val, Image< O > &output)`
Point-wise division of the value val to image input.
- `template<typename L , typename R >`
`void div_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise division of image rhs in image lhs.
- `template<typename L , typename R >`
`mln::trait::concrete< L >::ret min (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise min of images lhs and rhs.
- `template<typename L , typename R >`
`void min_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise min of image lhs in image rhs.
- `template<typename L , typename R >`
`mln::trait::op::minus< L, R >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.
- `template<typename L , typename R , typename F >`
`mln::trait::ch_value< L, typename F::result >::ret minus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F > &f)`
Point-wise addition of images lhs and rhs.
- `template<typename V , typename L , typename R >`
`mln::trait::ch_value< L, V >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.
- `template<typename I , typename V >`
`mln::trait::op::minus< I, V >::ret minus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.
- `template<typename I , typename V , typename F >`
`mln::trait::ch_value< I, typename F::result >::ret minus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`
Point-wise addition of the value val to image input.

- `template<typename I, typename V >`
`I & minus_cst_inplace (Image< I > &input, const V &val)`
Point-wise addition of the value `val` to image `input`.
- `template<typename L, typename R >`
`void minus_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image `rhs` in image `lhs`.
- `template<typename L, typename R, typename F >`
`mln::trait::ch_value< L, typename F::result >::ret plus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F > &f)`
Point-wise addition of images `lhs` and `rhs`.
- `template<typename V, typename L, typename R >`
`mln::trait::ch_value< L, V >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images `lhs` and `rhs`.
- `template<typename L, typename R >`
`mln::trait::op::plus< L, R >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images `lhs` and `rhs`.
- `template<typename I, typename V >`
`mln::trait::op::plus< I, V >::ret plus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value `val` to image `input`.
- `template<typename I, typename V, typename F >`
`mln::trait::ch_value< I, typename F::result >::ret plus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`
Point-wise addition of the value `val` to image `input`.
- `template<typename W, typename I, typename V >`
`mln::trait::ch_value< I, W >::ret plus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value `val` to image `input`.
- `template<typename I, typename V >`
`I & plus_cst_inplace (Image< I > &input, const V &val)`
Point-wise addition of the value `val` to image `input`.
- `template<typename L, typename R >`
`void plus_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image `rhs` in image `lhs`.
- `template<typename I >`
`mln::trait::concrete< I >::ret revert (const Image< I > &input)`
Point-wise reversion of image `input`.
- `template<typename I >`
`void revert_inplace (Image< I > &input)`
Point-wise in-place reversion of image `input`.

- `template<typename L , typename R , typename O >`
`void times (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`
Point-wise addition of images lhs and rhs.
- `template<typename I , typename V , typename O >`
`void times_cst (const Image< I > &input, const V &val, Image< O > &output)`
Point-wise addition of the value val to image input.
- `template<typename L , typename R >`
`void times_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image rhs in image lhs.

9.14.1 Detailed Description

Namespace of arithmetic.

9.14.2 Function Documentation

9.14.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::arith::diff_abs (const Image< I > & lhs, const Image< I > & rhs) [inline]`

Point-wise absolute difference of images lhs and rhs.

Parameters

- [in] *lhs* First operand image.
 [in] *rhs* Second operand image.

Returns

The result image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 63 of file arith/diff_abs.hh.

References `mln::data::transform()`.

9.14.2.2 `template<typename L , typename R , typename O > void mln::arith::div (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise division of images lhs and rhs.

Parameters

- [in] *lhs* First operand image.
 [in] *rhs* Second operand image.
 [out] *output* The result image.

Precondition

```
output.domain == lhs.domain == rhs.domain
```

Definition at line 227 of file arith/div.hh.

9.14.2.3 `template<typename I , typename V , typename O > void mln::arith::div_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise division of the value `val` to image `input`.

Parameters

[in] *input* The image.

[in] *val* The value.

[out] *output* The result image.

Precondition

```
output.domain == input.domain
```

Definition at line 242 of file arith/div.hh.

References `div_cst()`.

Referenced by `div_cst()`.

9.14.2.4 `template<typename L , typename R > void mln::arith::div_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise division of image `rhs` in image `lhs`.

Parameters

[in] *lhs* First operand image (subject to division).

[in, out] *rhs* Second operand image (to div `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) /= rhs(p)`

Precondition

```
rhs.domain <= lhs.domain
```

Definition at line 255 of file arith/div.hh.

References `div_inplace()`.

Referenced by `div_inplace()`.

9.14.2.5 `template<typename L , typename R > mln::trait::concrete< L >::ret mln::arith::min (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise min of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 112 of file `arith/min.hh`.

References `mln::initialize()`.

9.14.2.6 `template<typename L , typename R > void mln::arith::min_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise min of image `lhs` in image `rhs`.

Parameters

- [in, out] *lhs* First operand image.
- [in] *rhs* Second operand image.

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 128 of file `arith/min.hh`.

9.14.2.7 `template<typename L , typename R > mln::trait::op::minus< L, R >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 331 of file `arith/minus.hh`.

References `mln::initialize()`.

9.14.2.8 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.
- [in] *f* [Function](#).

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 350 of file `arith/minus.hh`.

References `mln::initialize()`.

9.14.2.9 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

Returns

The result image.

The free parameter `V` sets the destination value type.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 369 of file `arith/minus.hh`.

9.14.2.10 `template<typename I , typename V > mln::trait::op::minus< I, V >::ret mln::arith::minus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

- [in] *input* The image.

[in] *val* The value.

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 387 of file arith/minus.hh.

9.14.2.11 `template<typename I, typename V, typename F> mln::trait::ch_value< I, typename F::result >::ret mln::arith::minus_cst (const Image< I> & input, const V & val, const Function_v2v< F> & f) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in] *input* The image.

[in] *val* The value.

[in] *f* [Function](#).

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 405 of file arith/minus.hh.

9.14.2.12 `template<typename I, typename V> I & mln::arith::minus_cst_inplace (Image< I> & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in, out] *input* The image.

[in] *val* The value.

Precondition

`input.is_valid`

Definition at line 440 of file arith/minus.hh.

References `minus_cst_inplace()`, and `minus_inplace()`.

Referenced by `minus_cst_inplace()`.

9.14.2.13 `template<typename L , typename R > void mln::arith::minus_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image (subject to addition).
 [in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) -= rhs(p)`

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 424 of file `arith/minus.hh`.

References `minus_inplace()`.

Referenced by `minus_cst_inplace()`, and `minus_inplace()`.

9.14.2.14 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.
 [in] *rhs* Second operand image.
 [in] *f* [Function](#).

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 367 of file `arith/plus.hh`.

References `mln::initialize()`.

9.14.2.15 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
 [in] *rhs* Second operand image.

Returns

The result image.

The free parameter *V* sets the destination value type.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 386 of file arith/plus.hh.

9.14.2.16 `template<typename L , typename R > mln::trait::op::plus< L, R >::ret
 mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images *lhs* and *rhs*.

Parameters

- [in] *lhs* First operand image.
 [in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 348 of file arith/plus.hh.

References `mln::initialize()`.

9.14.2.17 `template<typename I , typename V > mln::trait::op::plus< I, V >::ret
 mln::arith::plus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

- [in] *input* The image.
 [in] *val* The value.

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 404 of file arith/plus.hh.

Referenced by `plus_cst()`.

9.14.2.18 `template<typename I , typename V , typename F > mln::trait::ch_value< I, typename F::result >::ret mln::arith::plus_cst (const Image< I > & input, const V & val, const Function_v2v< F > & f) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in] *input* The image.

[in] *val* The value.

[in] *f* [Function](#).

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 422 of file arith/plus.hh.

9.14.2.19 `template<typename W , typename I , typename V > mln::trait::ch_value< I, W >::ret mln::arith::plus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in] *input* The image.

[in] *val* The value.

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 441 of file arith/plus.hh.

References `plus_cst()`.

9.14.2.20 `template<typename I , typename V > I & mln::arith::plus_cst_inplace (Image< I > & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in, out] *input* The image.

[in] *val* The value.

Precondition

```
input.is_valid
```

Definition at line 475 of file arith/plus.hh.

References `plus_cst_inplace()`, and `plus_inplace()`.

Referenced by `plus_cst_inplace()`.

9.14.2.21 `template<typename L , typename R > void mln::arith::plus_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) += rhs(p)`

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 459 of file arith/plus.hh.

References `plus_inplace()`.

Referenced by `plus_cst_inplace()`, and `plus_inplace()`.

9.14.2.22 `template<typename I > mln::trait::concrete< I >::ret mln::arith::revert (const Image< I > & input) [inline]`

Point-wise reversion of image `input`.

Parameters

[in] *input* the input image.

Returns

The result image.

Precondition

```
input.is_valid
```

It performs:

for all `p` of `input.domain`

`output(p) = min + (max - input(p))`

Definition at line 158 of file revert.hh.

References `mln::initialize()`.

9.14.2.23 `template<typename I > void mln::arith::revert_inplace (Image< I > & input) [inline]`

Point-wise in-place reversion of image *input*.

Parameters

[in, out] *input* The target image.

Precondition

`input.is_valid`

It performs:

for all *p* of `input.domain`

`input(p) = min + (max - input(p))`

Definition at line 174 of file `revert.hh`.

9.14.2.24 `template<typename L , typename R , typename O > void mln::arith::times (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise addition of images *lhs* and *rhs*.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[out] *output* The result image.

Precondition

`output.domain == lhs.domain == rhs.domain`

Definition at line 226 of file `arith/times.hh`.

9.14.2.25 `template<typename I , typename V , typename O > void mln::arith::times_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in] *input* The image.

[in] *val* The value.

[out] *output* The result image.

Precondition

`output.domain == input.domain`

Definition at line 241 of file `arith/times.hh`.

References `times_cst()`.

Referenced by `times_cst()`.

9.14.2.26 `template<typename L , typename R > void mln::arith::times_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

- [in] *lhs* First operand image (subject to addition).
- [in, out] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) *= rhs(p)`

Precondition

`rhs.domain <= lhs.domain`

Definition at line 254 of file `arith/times.hh`.

References `times_inplace()`.

Referenced by `times_inplace()`.

9.15 `mln::arith::impl` Namespace Reference

Implementation namespace of `arith` namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of `arith` namespace.

9.15.1 Detailed Description

Implementation namespace of `arith` namespace.

9.16 `mln::arith::impl::generic` Namespace Reference

Generic implementation namespace of `arith` namespace.

9.16.1 Detailed Description

Generic implementation namespace of `arith` namespace.

9.17 mln::binarization Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I , typename F >
mln::trait::ch_value< I, bool >::ret binarization (const Image< I > &input, const Function_v2b< F
> &fun)`

Thresholds the values of input so that they can be stored in the output binary image.

- `template<typename I >
mln::trait::ch_value< I, bool >::ret threshold (const Image< I > &input, const typename I::value
threshold)`

Thresholds the values of input so that they can be stored in the output binary image.

9.17.1 Detailed Description

Namespace of "point-wise" expression tools.

9.17.2 Function Documentation

- 9.17.2.1** `template<typename I , typename F > mln::trait::ch_value< I, bool >::ret
mln::binarization::binarization (const Image< I > & input, const Function_v2b< F >
& fun) [inline]`

Thresholds the values of `input` so that they can be stored in the `output` binary image.

Parameters

[in] *input* The input image.

[in] *fun* The thresholding function, from value(I) to bool.

`for_all(p), output(p) = fun(p)`

Definition at line 86 of file `binarization.hh`.

Referenced by `threshold()`.

- 9.17.2.2** `template<typename I > mln::trait::ch_value< I, bool >::ret mln::binarization::threshold
(const Image< I > & input, const typename I::value threshold) [inline]`

Thresholds the values of `input` so that they can be stored in the `output` binary image.

Parameters

[in] *input* The input image.

[in] *threshold* The threshold.

If input(p) is greater or equal than the threshold, the value in the output image in the same point will be TRUE, else FALSE.

Definition at line 61 of file binarization/threshold.hh.

References `binarization()`.

9.18 mln::border Namespace Reference

Namespace of routines related to image virtual (outer) border.

Namespaces

- namespace `impl`
Implementation namespace of border namespace.

Functions

- `template<typename I >`
`void adjust (const Image< I > &ima, unsigned min_thickness)`
- `template<typename I >`
`void duplicate (const Image< I > &ima)`
- `template<typename I, typename J >`
`void equalize (const Image< I > &ima1, const Image< J > &ima2, unsigned min_thickness)`
- `template<typename I >`
`void fill (const Image< I > &ima, const typename I::value &v)`
- `template<typename I >`
`unsigned find (const Image< I > &ima)`
- `template<typename I >`
`unsigned get (const Image< I > &ima)`
- `template<typename I >`
`void mirror (const Image< I > &ima)`
- `template<typename I >`
`void resize (const Image< I > &ima, unsigned thickness)`

Facade.

9.18.1 Detailed Description

Namespace of routines related to image virtual (outer) border.

9.18.2 Function Documentation

9.18.2.1 `template<typename I > void mln::border::adjust (const Image< I > & ima, unsigned min_thickness) [inline]`

Adjust the virtual (outer) border of image `ima` so that its size is at least `min_thickness`.

Parameters

[in, out] *ima* The image whose border is to be adjusted.
[in] *min_thickness* The expected border minimum thickness.

Precondition

ima has to be initialized.

Warning

If the image border is already larger than *min_thickness*, this routine is a no-op.

Definition at line 62 of file border/adjust.hh.

References `get()`, and `resize()`.

9.18.2.2 template<typename I> void mln::border::duplicate (const Image< I> & *ima*)

Assign the virtual (outer) border of image *ima* with the duplicate of the inner border of this image.

Parameters

[in, out] *ima* The image whose border is to be duplicated.

Precondition

ima has to be initialized.

Definition at line 252 of file border/duplicate.hh.

References `get()`.

9.18.2.3 template<typename I, typename J> void mln::border::equalize (const Image< I> & *ima1*, const Image< J> & *ima2*, unsigned *min_thickness*) [inline]

Equalize the virtual (outer) border of images *ima1* and *ima2* so that their size is equal and is at least *min_thickness*.

Parameters

[in, out] *ima1* The first image whose border is to be equalized.
[in, out] *ima2* The second image whose border is to be equalized.
[in] *min_thickness* The expected border minimum thickness of both images.

Precondition

ima1 has to be initialized.
ima2 has to be initialized.

Warning

If both image borders already have the same thickness and if this thickness is larger than *min_thickness*, this routine is a no-op.

Definition at line 112 of file border/equalize.hh.

References `get()`.

9.18.2.4 `template<typename I> void mln::border::fill (const Image< I> & ima, const typename I::value & v) [inline]`

Fill the virtual (outer) border of image *ima* with the single value *v*.

Parameters

[in, out] *ima* The image whose border is to be filled.

[in] *v* The value to assign to all border pixels.

Precondition

ima has to be initialized.

Definition at line 201 of file border/fill.hh.

9.18.2.5 `template<typename I> unsigned mln::border::find (const Image< I> & ima) [inline]`

Find the virtual (outer) border thickness of image *ima*.

Parameters

[in] *ima* The image.

Returns

The border thickness (0 if there is no border).

Precondition

ima has to be initialized.

Definition at line 95 of file find.hh.

9.18.2.6 `template<typename I> unsigned mln::border::get (const Image< I> & ima) [inline]`

Get the virtual (outer) border thickness of image *ima*.

Parameters

[in] *ima* The image.

Returns

The border thickness (0 if there is no border).

Precondition

ima has to be initialized.

Definition at line 90 of file border/get.hh.

Referenced by `adjust()`, `duplicate()`, and `equalize()`.

9.18.2.7 `template<typename I> void mln::border::mirror (const Image< I> & ima) [inline]`

Mirror the virtual (outer) border of image *ima* with the (inner) level contents of this image.

Parameters

[in, out] *ima* The image whose border is to be mirrored.

Precondition

ima has to be initialized.

Definition at line 211 of file border/mirror.hh.

9.18.2.8 `template<typename I> void mln::border::resize (const Image< I> & ima, unsigned thickness) [inline]`

Facade.

Resize the virtual (outer) border of image *ima* to exactly *thickness*.

Parameters

[in, out] *ima* The image whose border is to be resized.

[in] *thickness* The expected border thickness.

Precondition

ima has to be initialized.

Warning

If the image border already has the expected thickness, this routine is a no-op.

Definition at line 126 of file resize.hh.

References `mln::primary()`, and `resize()`.

Referenced by `adjust()`, and `resize()`.

9.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of border namespace.

9.19.1 Detailed Description

Implementation namespace of border namespace.

9.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

9.20.1 Detailed Description

Generic implementation namespace of border namespace.

9.21 mln::canvas Namespace Reference

Namespace of canvas.

Namespaces

- namespace [browsing](#)
Namespace of browsing canvas.
- namespace [impl](#)
Implementation namespace of canvas namespace.
- namespace [labeling](#)
Namespace of labeling canvas.
- namespace [morpho](#)
Namespace of morphological canvas.

Classes

- struct [chamfer](#)
Compute chamfer distance.

Functions

- template<typename I , typename N , typename W , typename D , typename F >
mln::trait::ch_value< I, D >::ret [distance_front](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh, const [Weighted_Window](#)< W > &w_win, D max, F &functor)
Canvas of discrete distance computation by thick front propagation.
- template<typename I , typename N , typename D , typename F >
mln::trait::ch_value< I, D >::ret [distance_geodesic](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh, D max, F &functor)
Discrete geodesic distance canvas.

9.21.1 Detailed Description

Namespace of canvas.

9.21.2 Function Documentation

9.21.2.1 `template<typename I , typename N , typename W , typename D , typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max, F & functor) [inline]`

Canvas of discrete distance computation by thick front propagation.

Definition at line 397 of file canvas/distance_front.hh.

Referenced by `mln::transform::influence_zone_front()`.

9.21.2.2 `template<typename I , typename N , typename D , typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max, F & functor) [inline]`

Discrete geodesic distance canvas.

Definition at line 321 of file canvas/distance_geodesic.hh.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`.

9.22 mln::canvas::browsing Namespace Reference

Namespace of browsing canvas.

Classes

- struct [backdiagonal2d_t](#)
Browsing in a certain direction.
- struct [breadth_first_search_t](#)
Breadth-first search algorithm for graph, on vertices.
- struct [depth_first_search_t](#)
Breadth-first search algorithm for graph, on vertices.
- struct [diagonal2d_t](#)
Browsing in a certain direction.
- struct [dir_struct_elt_incr_update_t](#)
Browsing in a certain direction with a segment.
- struct [directional_t](#)
Browsing in a certain direction.

- struct [fwd_t](#)
Canvas for forward browsing.
- struct [hyper_directional_t](#)
Browsing in a certain direction.
- struct [snake_fwd_t](#)
Browsing in a snake-way, forward.
- struct [snake_generic_t](#)
Multidimensional Browsing in a given-way.
- struct [snake_vert_t](#)
Browsing in a snake-way, forward.

9.22.1 Detailed Description

Namespace of browsing canvas.

9.23 mln::canvas::impl Namespace Reference

Implementation namespace of canvas namespace.

9.23.1 Detailed Description

Implementation namespace of canvas namespace.

9.24 mln::canvas::labeling Namespace Reference

Namespace of labeling canvas.

Namespaces

- namespace [impl](#)
Implementation namespace of labeling canvas namespace.

Functions

- `template<typename I , typename N , typename L , typename F >
mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input_, const Neighborhood< N >
&nbh_, L &nlabels, F &functor)`
Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

9.24.1 Detailed Description

Namespace of labeling canvas.

9.24.2 Function Documentation

9.24.2.1 `template<typename I , typename N , typename L , typename F > mln::trait::ch_value< I, L >::ret mln::canvas::labeling::blobs (const Image< I > & input_, const Neighborhood< N > & nbh_, L & nlabels, F & functor) [inline]`

Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the objects.
- [out] *nlabels* The Number of labels. Its value is set in the algorithms.
- [in, out] *functor* A functor computing data while labeling.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 167 of file canvas/labeling/blobs.hh.

9.25 mln::canvas::labeling::impl Namespace Reference

Implementation namespace of labeling canvas namespace.

9.25.1 Detailed Description

Implementation namespace of labeling canvas namespace.

9.26 mln::canvas::morpho Namespace Reference

Namespace of morphological canvas.

9.26.1 Detailed Description

Namespace of morphological canvas.

9.27 mln::convert Namespace Reference

Namespace of conversion routines.

Functions

- template<typename V >
void [from_to](#) (const float &from, [Value](#)< V > &to)
Conversion of a float `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const int &from, [Value](#)< V > &to)
Conversion of a int `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const double &from, [Value](#)< V > &to)
Conversion of a double `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const unsigned &from, [Value](#)< V > &to)
Conversion of an unsigned `from` towards a value `to`.
- template<typename S >
[mln_image_from_grid](#) (typename S::site::grid, bool) to_image(const [Site_Set](#)< S > &pset)
Convert a point set `pset` into a binary image.
- template<typename W >
[mln_image_from_grid](#) (typename W::site::grid, bool) to_image(const [Window](#)< W > &win)
Convert a window `win` into a binary image.
- template<typename W >
[mln_image_from_grid](#) (typename W::site::grid, [mln_weight](#)(W)) to_image(const [Weighted_Window](#)< W > &w_win)
Convert a weighted window `w_win` into an image.
- template<typename N >
[mln_image_from_grid](#) (typename N::site::grid, bool) to_image(const [Neighborhood](#)< N > &nbh)
Convert a neighborhood `nbh` into a binary image.
- template<typename N >
[mln_window](#) (N) to_window(const [Neighborhood](#)< N > &nbh)
Convert a neighborhood `nbh` into a window.
- template<typename T , typename O >
[T to](#) (const O &from)
Conversion of the object `from` towards an object with type `T`.
- template<typename P >
[P::dpoint to_dpoint](#) (const [Point_Site](#)< P > &p)
Convert a point site `p` into a delta-point.

- `template<typename I >`
`pw::value_< I > to_fun (const Image< I > &ima)`
Convert an image into a function.
- `template<typename T >`
`imageId< unsigned > to_image (const histo::array< T > &h)`
Convert an histo h into an imageId<unsigned>.
- `template<typename I >`
`p_array< typename I::psite > to_p_array (const Image< I > &img)`
Convert an image img into a p_array.
- `template<typename S >`
`p_array< typename S::psite > to_p_array (const Site_Set< S > &pset)`
Convert a point set pset into a p_array (point set vector).
- `template<typename W >`
`p_array< typename W::psite > to_p_array (const Window< W > &win, const typename W::psite &p)`
Convert a window win centered at point p into a p_array (point set vector).
- `template<typename N >`
`p_set< typename N::psite > to_p_set (const Neighborhood< N > &nbh)`
Convert a neighborhood nbh into a site set.
- `template<typename I >`
`p_set< typename I::psite > to_p_set (const Image< I > &ima)`
Convert a binary image ima into a site set.
- `template<typename P , typename C >`
`p_set< P > to_p_set (const std::set< P, C > &s)`
Convert an std::set s of sites into a site set.
- `template<typename S >`
`p_set< typename S::psite > to_p_set (const Site_Set< S > &ps)`
Convert any site set ps into a 'mlnp_set' site set.
- `template<typename W >`
`p_set< typename W::psite > to_p_set (const Window< W > &win)`
Convert a Window win into a site set.
- `template<typename I >`
`QImage to_qimage (const Image< I > &ima)`
Convert a Milena image to a QImage.
- `template<typename N >`
`window< typename N::dpoint > to_upper_window (const Neighborhood< N > &nbh)`
Convert a neighborhood nbh into an upper window.

- `template<typename W >`
`window< typename W::dpsite > to_upper_window (const Window< W > &win)`
Convert a window `nbh` into an upper window.
- `template<typename D , typename C >`
`window< D > to_window (const std::set< D, C > &s)`
Convert an `std::set` `s` of delta-sites into a window.
- `template<typename I >`
`window< typename I::site::dpsite > to_window (const Image< I > &ima)`
Convert a binary image `ima` into a window.
- `template<typename S >`
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`
Convert a site set `pset` into a window.

Variables

- `fun::C< R(*) (A)> to_fun (R(*f)(A))`
Convert a C unary function into an `mln::fun::C`.

9.27.1 Detailed Description

Namespace of conversion routines.

9.27.2 Function Documentation

9.27.2.1 `template<typename V > void mln::convert::from_to (const float & from, Value< V > & to)`

Conversion of a float `from` towards a value `to`.

9.27.2.2 `template<typename V > void mln::convert::from_to (const int & from, Value< V > & to)`

Conversion of a int `from` towards a value `to`.

9.27.2.3 `template<typename V > void mln::convert::from_to (const double & from, Value< V > & to)`

Conversion of a double `from` towards a value `to`.

9.27.2.4 `template<typename V > void mln::convert::from_to (const unsigned & from, Value< V > & to)`

Conversion of an unsigned `from` towards a value `to`.

9.27.2.5 `template<typename S> mln::convert::mln_image_from_grid (typename S::site::grid, bool) const [inline]`

Convert a point set `pset` into a binary image.

Width of the converted image will be `pset.bbox + 2 * border`.

9.27.2.6 `template<typename W> mln::convert::mln_image_from_grid (typename W::site::grid, bool) const`

Convert a window `win` into a binary image.

9.27.2.7 `template<typename W> mln::convert::mln_image_from_grid (typename W::site::grid, mln_weight(W)) const`

Convert a weighted window `w_win` into an image.

9.27.2.8 `template<typename N> mln::convert::mln_image_from_grid (typename N::site::grid, bool) const`

Convert a neighborhood `nbh` into a binary image.

9.27.2.9 `template<typename N> mln::convert::mln_window (N) const [inline]`

Convert a neighborhood `nbh` into a window.

Definition at line 74 of file `to_window.hh`.

9.27.2.10 `template<typename T, typename O> T mln::convert::to (const O & from) [inline]`

Conversion of the object `from` towards an object with type `T`.

Definition at line 63 of file `to.hh`.

References `mln::mln_exact()`.

Referenced by `mln::make_debug_graph_image()`.

9.27.2.11 `template<typename P> P::dpoint mln::convert::to_dpoint (const Point_Site< P> & p) [inline]`

Convert a point site `p` into a delta-point.

Definition at line 52 of file `to_dpoint.hh`.

9.27.2.12 `template<typename I> pw::value_<I> mln::convert::to_fun (const Image< I> & ima)`

Convert an image into a function.

Definition at line 64 of file `to_fun.hh`.

9.27.2.13 `template<typename T> image1d<unsigned> mln::convert::to_image (const histo::array< T> & h)`

Convert an `histo h` into an `image1d<unsigned>`.

9.27.2.14 `template<typename I> p_array< typename I::psite> mln::convert::to_p_array (const Image< I> & img) [inline]`

Convert an image `img` into a `p_array`.

Definition at line 98 of file `to_p_array.hh`.

References `mln::p_array< P>::append()`.

9.27.2.15 `template<typename S> p_array< typename S::psite> mln::convert::to_p_array (const Site_Set< S> & pset) [inline]`

Convert a point set `pset` into a `p_array` (point set vector).

Definition at line 68 of file `to_p_array.hh`.

References `mln::p_array< P>::append()`.

9.27.2.16 `template<typename W> p_array< typename W::psite> mln::convert::to_p_array (const Window< W> & win, const typename W::psite & p) [inline]`

Convert a window `win` centered at point `p` into a `p_array` (point set vector).

Definition at line 82 of file `to_p_array.hh`.

References `mln::p_array< P>::append()`, and `mln::p_array< P>::reserve()`.

9.27.2.17 `template<typename N> p_set< typename N::psite> mln::convert::to_p_set (const Neighborhood< N> & nbh) [inline]`

Convert a neighborhood `nbh` into a site set.

Definition at line 83 of file `to_p_set.hh`.

References `mln::p_set< P>::insert()`.

9.27.2.18 `template<typename I> p_set< typename I::psite> mln::convert::to_p_set (const Image< I> & ima) [inline]`

Convert a binary image `ima` into a site set.

Definition at line 97 of file `to_p_set.hh`.

References `mln::p_set< P>::insert()`.

9.27.2.19 `template<typename P, typename C> p_set< P> mln::convert::to_p_set (const std::set< P, C> & s) [inline]`

Convert an `std::set s` of sites into a site set.

C is the comparison functor.

Definition at line 130 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.20 `template<typename S > p_set< typename S::psite > mln::convert::to_p_set (const Site_Set< S > & ps) [inline]`

Convert any site set `ps` into a 'mlnp_set' site set.

Definition at line 144 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.21 `template<typename W > p_set< typename W::psite > mln::convert::to_p_set (const Window< W > & win) [inline]`

Convert a [Window](#) `win` into a site set.

Definition at line 117 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.22 `template<typename I > QImage mln::convert::to_qimage (const Image< I > & ima) [inline]`

Convert a Milena image to a QImage.

Definition at line 279 of file to_qimage.hh.

9.27.2.23 `template<typename N > window< typename N::dpoint > mln::convert::to_upper_window (const Neighborhood< N > & nbh) [inline]`

Convert a neighborhood `nbh` into an upper window.

Definition at line 82 of file to_upper_window.hh.

References mln::window< D >::insert().

9.27.2.24 `template<typename W > window< typename W::dpsite > mln::convert::to_upper_window (const Window< W > & win) [inline]`

Convert a window `nbh` into an upper window.

Definition at line 67 of file to_upper_window.hh.

References mln::window< D >::insert().

9.27.2.25 `template<typename D, typename C > window< D > mln::convert::to_window (const std::set< D, C > & s) [inline]`

Convert an `std::set` `s` of delta-sites into a window.

Definition at line 128 of file to_window.hh.

References `mln::window< D >::insert()`.

9.27.2.26 `template<typename I > window< typename I::site::dpsite > mln::convert::to_window (const Image< I > & ima) [inline]`

Convert a binary image `ima` into a window.

Definition at line 94 of file `to_window.hh`.

References `mln::window< D >::insert()`.

Referenced by `to_window()`.

9.27.2.27 `template<typename S > window< typename S::site::dpsite > mln::convert::to_window (const Site_Set< S > & pset) [inline]`

Convert a site set `pset` into a window.

Definition at line 117 of file `to_window.hh`.

References `to_window()`.

9.27.3 Variable Documentation

9.27.3.1 `pw::value_< I > mln::convert::to_fun [inline]`

Convert a C unary function into an `mln::fun::C`.

Definition at line 45 of file `to_fun.hh`.

9.28 mln::data Namespace Reference

Namespace of image processing routines related to pixel data.

Namespaces

- namespace [approx](#)
Namespace of image processing routines related to pixel levels with approximation.
- namespace [impl](#)
Implementation namespace of data namespace.
- namespace [naive](#)
Namespace of image processing routines related to pixel levels with naive approach.

Functions

- `template<typename I , typename O > void abs (const Image< I > &input, Image< O > &output)`

- template<typename I >
void [abs_inplace](#) (Image< I > &input)
- template<typename I, typename F >
void [apply](#) (Image< I > &input, const [Function_v2v](#)< F > &f)
- template<typename A, typename I >
A::result [compute](#) (const [Accumulator](#)< A > &a, const Image< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename A, typename I >
A::result [compute](#) ([Accumulator](#)< A > &a, const Image< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename V, typename I >
mln::trait::ch_value< I, V >::ret [convert](#) (const V &v, const Image< I > &input)
Convert the image input by changing the value type.
- template<typename I, typename W, typename O >
void [fast_median](#) (const Image< I > &input, const [Window](#)< W > &win, Image< O > &output)
- template<typename I, typename D >
void [fill](#) (Image< I > &ima, const D &data)
- template<typename I, typename J >
void [fill_with_image](#) (Image< I > &ima, const Image< J > &data)
Fill the image ima with the values of the image data.
- template<typename I, typename W >
mln::trait::concrete< I >::ret [median](#) (const Image< I > &input, const [Window](#)< W > &win)
- template<typename A, typename I >
[mln_meta_accu_result](#) (A, typename I::value) compute(const [Meta_Accumulator](#)< A > &a
Compute an accumulator onto the pixel values of the image input.
- template<typename I, typename J >
void [paste](#) (const Image< I > &input, Image< J > &output)
Paste the contents of image input into the image output.
- template<typename I, typename J >
void [paste_without_localization](#) (const Image< I > &input, Image< J > &output)
Paste the contents of image input into the image output without taking into account the localization of sites.
- template<typename I >
void [replace](#) (Image< I > &input, const typename I::value &old_value, const typename I::value &new_value)
- template<typename I, typename V >
mln::trait::ch_value< I, V >::ret [saturate](#) (const Image< I > &input, const V &min, const V &max)
- template<typename V, typename I >
mln::trait::ch_value< I, V >::ret [saturate](#) (V v, const Image< I > &input)
- template<typename I >
void [saturate_inplace](#) (Image< I > &input, const typename I::value &min, const typename I::value &max)
- template<typename I >
[util::array](#)< unsigned > [sort_offsets_increasing](#) (const Image< I > &input)

Sort pixel offsets of the image `input` wrt increasing pixel values.

- `template<typename I >`
`p_array< typename I::psite > sort_psites_decreasing (const Image< I > &input)`
- `template<typename I >`
`p_array< typename I::psite > sort_psites_increasing (const Image< I > &input)`
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
- `template<typename I , typename O >`
`void to_enc (const Image< I > &input, Image< O > &output)`
- `template<typename I1 , typename I2 , typename F >`
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1, const Image< I2 > &input2, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input, const Function_v2v< F > &f)`
- `template<typename I1 , typename I2 , typename F >`
`void transform_inplace (Image< I1 > &ima, const Image< I2 > &aux, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`
`void transform_inplace (Image< I > &ima, const Function_v2v< F > &f)`
- `template<typename A , typename I >`
`A::result update (Accumulator< A > &a, const Image< I > &input)`
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret wrap (const V &v, const Image< I > &input)`

Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

- `template<typename I , typename V >`
`void fill_with_value (Image< I > &ima, const V &val)`
Fill the whole image `ima` with the single value `v`.

9.28.1 Detailed Description

Namespace of image processing routines related to pixel data.

9.28.2 Function Documentation

9.28.2.1 `template<typename I , typename O > void mln::data::abs (const Image< I > & input, Image< O > & output) [inline]`

Apply the absolute value (abs) function to image pixel values.

Parameters

- [in] *input* The input image.
[out] *output* The output image.

Definition at line 68 of file `data/abs.hh`.

References `transform()`.

9.28.2.2 `template<typename I > void mln::data::abs_inplace (Image< I > & input) [inline]`

Apply the absolute value (abs) function to image pixel values.

Parameters

[in, out] *input* The input image.

Definition at line 80 of file data/abs.hh.

References `apply()`.

9.28.2.3 `template<typename I , typename F > void mln::data::apply (Image< I > & input, const Function_v2v< F > & f) [inline]`

Apply a function-object to the image `input`.

Parameters

[in, out] *input* The input image.

[in] *f* The function-object.

This routine runs:

for all `p` of `input`, `input (p) = f(input (p))`

This routine is equivalent to `data::transform(input, f, input)` but it is faster since a single iterator is required.

Definition at line 95 of file `apply.hh`.

Referenced by `abs_inplace()`, and `saturate_inplace()`.

9.28.2.4 `template<typename A , typename I > A::result mln::data::compute (const Accumulator< A > & a, const Image< I > & input) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Be ware that the given accumulator won't be modified and won't store any result.

Parameters

[in] *a* An accumulator.

[in] *input* The input image.

Returns

The accumulator result.

It fully relies on [data::update](#).

Definition at line 93 of file `data/compute.hh`.

Referenced by `mln::labeled_image< I >::labeled_image()`, `mln::estim::mean()`, `mln::estim::min_max()`, and `mln::estim::sum()`.

9.28.2.5 `template<typename A , typename I > A::result mln::data::compute (Accumulator< A > & a, const Image< I > & input) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Parameters

[in, out] *a* An accumulator.

[in] *input* The input image.

Returns

The accumulator result.

It fully relies on [data::update](#).

Definition at line 104 of file `data/compute.hh`.

9.28.2.6 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::convert (const V & v, const Image< I > & input) [inline]`

Convert the image `input` by changing the value type.

Parameters

[in] *v* A value of the destination type.

[in] *input* The input image.

Definition at line 154 of file `mln/data/convert.hh`.

Referenced by `mln::morpho::watershed::superpose()`, and `mln::debug::superpose()`.

9.28.2.7 `template<typename I , typename W , typename O > void mln::data::fast_median (const Image< I > & input, const Window< W > & win, Image< O > & output) [inline]`

Compute in `output` the median filter of image `input` by the window `win`.

Parameters

[in] *input* The image to be filtered.

[in] *win* The window.

[in, out] *output* The output image.

Precondition

`input` and `output` have to be initialized.

Definition at line 167 of file `fast_median.hh`.

9.28.2.8 `template<typename I, typename D> void mln::data::fill (Image< I > & ima, const D & data) [inline]`

Fill the whole image `ima` with the data provided by `aux`.

Parameters

- `[in, out]` ***ima*** The image to be filled.
- `[in]` ***data*** The auxiliary data to fill the image `ima`.

Precondition

`ima` has to be initialized.

Definition at line 138 of file `data/fill.hh`.

Referenced by `mln::draw::box_plain()`, `mln::draw::dashed_line()`, `mln::topo::detach()`, `mln::util::display_branch()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::duplicate()`, `mln::make::edge_image()`, `mln::morpho::tree::filter::filter()`, `mln::transform::hough()`, `mln::registration::icp()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::graph::labeling()`, `mln::morpho::laplacian()`, `mln::make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::debug::mosaic()`, `mln::debug::slices_2d()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::morpho::watershed::topological()`, and `mln::geom::translate()`.

9.28.2.9 `template<typename I, typename J> void mln::data::fill_with_image (Image< I > & ima, const Image< J > & data) [inline]`

Fill the image `ima` with the values of the image `data`.

Parameters

- `[in, out]` ***ima*** The image to be filled.
- `[in]` ***data*** The image.

Warning

The definition domain of `ima` has to be included in the one of `data`.

Precondition

`ima.domain <= data.domain`.

Definition at line 124 of file `fill_with_image.hh`.

9.28.2.10 `template<typename I, typename V> void mln::data::fill_with_value (Image< I > & ima, const V & val) [inline]`

Fill the whole image `ima` with the single value `v`.

Parameters

- `[in, out]` ***ima*** The image to be filled.
- `[in]` ***val*** The value to assign to all sites.

Precondition

`ima` has to be initialized.

Definition at line 130 of file `fill_with_value.hh`.

Referenced by `mln::p_image< I >::clear()`.

9.28.2.11 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::median (const Image< I > & input, const Window< W > & win)`

Compute in `output` the median filter of image `input` by the window `win`.

Parameters

[in] *input* The image to be filtered.

[in] *win* The window.

Precondition

`input` have to be initialized.

Definition at line 270 of file `median.hh`.

9.28.2.12 `template<typename A , typename I > mln::data::mln_meta_accu_result (A , typename I::value) const [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Parameters

[in] *a* A meta-accumulator.

[in] *input* The input image.

Returns

The accumulator result.

9.28.2.13 `template<typename I , typename J > void mln::data::paste (const Image< I > & input, Image< J > & output) [inline]`

Paste the contents of image `input` into the image `output`.

Parameters

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

This routine runs:

for all `p` of `input`, `output (p) = input (p)`.

Warning

The definition domain of `input` has to be included in the one of `output`; so using `mln::safe_image` does not make pasting outside the output domain work.

Precondition

```
input.domain <= output.domain
```

Definition at line 137 of file `paste.hh`.

Referenced by `mln::make::image3d()`, `mln::draw::line()`, `mln::debug::mosaic()`, `mln::geom::rotate()`, `mln::debug::slices_2d()`, and `mln::labeling::superpose()`.

9.28.2.14 `template<typename I , typename J > void mln::data::paste_without_localization (const Image< I > & input, Image< J > & output) [inline]`

Paste the contents of image `input` into the image `output` without taking into account the localization of sites.

Parameters

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

Definition at line 349 of file `paste_without_localization.hh`.

9.28.2.15 `template<typename I > void mln::data::replace (Image< I > & input, const typename I::value & old_value, const typename I::value & new_value)`

Replace `old_value` by `new_value` in the image `input`

Parameters

[in] *input* The input image.

[in] *old_value* The value to be replaced...

[in] *new_value* ...by this one.

Definition at line 88 of file `replace.hh`.

9.28.2.16 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::saturate (V v, const Image< I > & input) [inline]`

Apply the saturate function to image pixel values.

Parameters

[in] *v* A value of the output type.

[in] *input* The input image.

The saturation is based on the min and max values of the output value type. This assumes that the range of values in the input image is larger than the one of the output image.

Definition at line 87 of file `data/saturate.hh`.

References `transform()`.

9.28.2.17 `template<typename I , typename V > mln::trait::ch_value< I, V >::ret
mln::data::saturate (const Image< I > & input, const V & min, const V & max)
[inline]`

Apply the saturate function to image pixel values.

Parameters

- [in] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value.

Definition at line 103 of file data/saturate.hh.

References transform().

9.28.2.18 `template<typename I > void mln::data::saturate_inplace (Image< I > & input, const
typename I::value & min, const typename I::value & max) [inline]`

Apply the saturate function to image pixel values.

Parameters

- [in, out] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value

Definition at line 119 of file data/saturate.hh.

References apply().

9.28.2.19 `template<typename I > util::array< unsigned > mln::data::sort_offsets_increasing (
const Image< I > & input) [inline]`

Sort pixel offsets of the image *input* wrt increasing pixel values.

Definition at line 298 of file sort_offsets.hh.

9.28.2.20 `template<typename I > p_array< typename I::psite > mln::data::sort_psites_
decreasing (const Image< I > & input) [inline]`

Sort psites the image *input* through a function *f* to set the *output* image in decreasing way.

Parameters

- [in] *input* The input image.

Precondition

`input.is_valid`

Definition at line 229 of file sort_psites.hh.

Referenced by mln::morpho::tree::min_tree().

9.28.2.21 `template<typename I > p_array< typename I::psite > mln::data::sort_psites_increasing (const Image< I > & input) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in increasing way.

Parameters

[in] *input* The input image.

Precondition

`input.is_valid`

Definition at line 219 of file `sort_psites.hh`.

Referenced by `mln::morpho::tree::max_tree()`.

9.28.2.22 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::stretch (const V & v, const Image< I > & input) [inline]`

Stretch the values of `input` so that they can be stored in `output`.

Parameters

[in] *v* A value to set the output value type.

[in] *input* The input image.

Returns

A stretch image with values of the same type as *v*.

Precondition

`input.is_valid`

Definition at line 128 of file `stretch.hh`.

References `mln::data::impl::stretch()`.

9.28.2.23 `template<typename I , typename O > void mln::data::to_enc (const Image< I > & input, Image< O > & output) [inline]`

Set the `output` image with the encoding values of the image `input` pixels.

Parameters

[in] *input* The input image.

[out] *output* The result image.

Precondition

`output.domain >= input.domain`

Definition at line 60 of file `data/to_enc.hh`.

References `transform()`.

9.28.2.24 `template<typename I , typename F > mln::trait::ch_value< I, typename F::result >::ret mln::data::transform (const Image< I > & input, const Function_v2v< F > & f) [inline]`

Transform the image *input* through a function *f*.

Parameters

[in] *input* The input image.

[in] *f* The function.

This routine runs:

for all *p* of *input*, *output* (*p*) = *f*(*input* (*p*)).

Definition at line 202 of file *data/transform.hh*.

Referenced by *abs()*, *mln::logical::and_not()*, *mln::labeling::colorize()*, *mln::arith::diff_abs()*, *mln::labeling::fill_holes()*, *mln::linear::mln_ch_convolve()*, *mln::linear::mln_ch_convolve_grad()*, *mln::labeling::pack()*, *mln::labeling::pack_inplace()*, *mln::labeling::relabel()*, *saturate()*, *mln::data::impl::stretch()*, *to_enc()*, *mln::labeling::wrap()*, and *wrap()*.

9.28.2.25 `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1, typename F::result >::ret mln::data::transform (const Image< I1 > & input1, const Image< I2 > & input2, const Function_vv2v< F > & f) [inline]`

Transform two images *input1* *input2* through a function *f*.

Parameters

[in] *input1* The 1st input image.

[in] *input2* The 2nd input image.

[in] *f* The function.

This routine runs:

for all *p* of *input*, *output* (*p*) = *f*(*input1* (*p*) , *input2* (*p*)).

Definition at line 219 of file *data/transform.hh*.

9.28.2.26 `template<typename I1 , typename I2 , typename F > void mln::data::transform_inplace (Image< I1 > & ima, const Image< I2 > & aux, const Function_vv2v< F > & f)`

Transform inplace the image *ima* with the image *aux* through a function *f*.

Parameters

[in] *ima* The image to be transformed.

[in] *aux* The auxiliary image.

[in] *f* The function.

This routine runs:

for all *p* of *ima*, *ima* (*p*) = *f*(*ima* (*p*) , *aux* (*p*)).

Definition at line 502 of file *transform_inplace.hh*.

9.28.2.27 `template<typename I , typename F > void mln::data::transform_inplace (Image< I > & ima, const Function_v2v< F > & f)`

Transform inplace the image *ima* through a function *f*.

Parameters

[in, out] *ima* The image to be transformed.

[in] *f* The function.

This routine runs:

for all *p* of *ima*, *ima* (*p*) = *f*(*ima* (*p*)).

Definition at line 490 of file transform_inplace.hh.

Referenced by `mln::logical::and_inplace()`, `mln::logical::and_not_inplace()`, `mln::logical::not_inplace()`, `mln::logical::or_inplace()`, `mln::labeling::relabel_inplace()`, and `mln::logical::xor_inplace()`.

9.28.2.28 `template<typename A , typename I > A::result mln::data::update (Accumulator< A > & a, const Image< I > & input) [inline]`

Update an accumulator with the pixel values of the image *input*.

Parameters

[in] *a* The accumulator.

[in] *input* The input image.

Returns

The accumulator result.

Definition at line 191 of file update.hh.

9.28.2.29 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::wrap (const V & v, const Image< I > & input)`

Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Parameters

[in] *v* The target value type.

[in] *input* Input image.

Returns

An image with wrapped values.

Definition at line 65 of file data/wrap.hh.

References `transform()`.

9.29 mln::data::approx Namespace Reference

Namespace of image processing routines related to pixel levels with approximation.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::approx](#) namespace.

Functions

- `template<typename I>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const win::rectangle2d &win)`
- `template<typename I>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const win::octagon2d &win)`
- `template<typename I>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const win::disk2d &win)`

9.29.1 Detailed Description

Namespace of image processing routines related to pixel levels with approximation.

9.29.2 Function Documentation

9.29.2.1 `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::rectangle2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D rectangle `win`.

Parameters

- [in] *input* The image to be filtered.
[in] *win* The rectangle.

The approximation is based on a vertical median ran after an horizontal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 112 of file `approx/median.hh`.

Referenced by `median()`.

9.29.2.2 `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::octagon2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D octagon `win`.

Parameters

- [in] *input* The image to be filtered.
- [in] *win* The octagon.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 159 of file `approx/median.hh`.

References `median()`.

9.29.2.3 `template<typename I> mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I> & input, const win::disk2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D disk `win`.

Parameters

- [in] *input* The image to be filtered.
- [in] *win* The disk.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 132 of file `approx/median.hh`.

References `median()`.

9.30 mln::data::approx::impl Namespace Reference

Implementation namespace of [data::approx](#) namespace.

9.30.1 Detailed Description

Implementation namespace of [data::approx](#) namespace.

9.31 mln::data::impl Namespace Reference

Implementation namespace of `data` namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of data namespace.

Functions

- `template<typename I , typename J >`
`void paste_without_localization_fast (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a point-wise copy.
- `template<typename I , typename J >`
`void paste_without_localization_fastest (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a one-block memory copy.
- `template<typename I , typename J >`
`void paste_without_localization_lines (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a line-per-line memory copy.
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input_)`
Generic implementation of [data::stretch](#).
- `template<typename I , typename F >`
`void transform_inplace_lowq (Image< I > &input_, const Function_v2v< F > &f_)`
Specialized implementation.
- `template<typename A , typename I >`
`A::result update_fastest (Accumulator< A > &a_, const Image< I > &input_)`
Fastest implementation of [data::update](#).

9.31.1 Detailed Description

Implementation namespace of data namespace.

9.31.2 Function Documentation

9.31.2.1 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fast (const Image< I > & input_, Image< J > & output_)` `[inline]`

Paste data to an image without using localization. Performs a point-wise copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.

Definition at line 220 of file `paste_without_localization.hh`.

9.31.2.2 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fastest (const Image< I > & input_, Image< J > & output_) [inline]`

Paste data to an image without using localization. Performs a one-block memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same border size.
- Same domain size.
- Same value type.

Definition at line 142 of file `paste_without_localization.hh`.

9.31.2.3 `template<typename I , typename J > void mln::data::impl::paste_without_localization_lines (const Image< I > & input_, Image< J > & output_) [inline]`

Paste data to an image without using localization. Performs a line-per-line memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.
- Same value type.

Definition at line 179 of file `paste_without_localization.hh`.

9.31.2.4 `template<typename V , typename I > mln::trait::ch_value< I , V >::ret
mln::data::impl::stretch (const V & v, const Image< I > & input) [inline]`

Generic implementation of [data::stretch](#).

Parameters

- [in] *v* A value to set the output value type.
- [in] *input* The input image.

Returns

A stretch image with values of the same type as *v*.

Definition at line 83 of file `stretch.hh`.

References `mln::initialize()`, `mln::estim::min_max()`, and `mln::data::transform()`.

Referenced by `mln::data::stretch()`.

9.31.2.5 `template<typename I , typename F > void mln::data::impl::transform_inplace_lowq (
Image< I > & input_, const Function_v2v< F > & f_)`

Specialized implementation.

Definition at line 203 of file `transform_inplace.hh`.

9.31.2.6 `template<typename A , typename I > A ::result mln::data::impl::update_fastest (
Accumulator< A > & a_, const Image< I > & input_) [inline]`

Fastest implementation of [data::update](#).

Parameters

- [in] *a_* The accumulator.
- [in] *input_* The input image.

Returns

The accumulator result.

Definition at line 129 of file `update.hh`.

9.32 mln::data::impl::generic Namespace Reference

Generic implementation namespace of data namespace.

Functions

- `template<typename I , typename J >
void fill_with_image (Image< I > &ima_, const Image< J > &data_)
Generic implementation.`

- `template<typename I , typename V >`
`void fill_with_value (Image< I > &ima_, const V &val)`
Fill the whole image `ima_` with the single value `v`.
- `template<typename I , typename J >`
`void paste (const Image< I > &input_, Image< J > &output_)`
Generic implementation of `data::paste`.
- `template<typename I , typename F >`
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input_, const Function_v2v< F > &f_)`
Generic implementation of `data::transform`.
- `template<typename I1 , typename I2 , typename F >`
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1_, const Image< I2 > &input2_, const Function_vv2v< F > &f_)`
Generic implementation of `data::transform`.
- `template<typename I1 , typename I2 , typename F >`
`void transform_inplace (Image< I1 > &ima_, const Image< I2 > &aux_, const Function_vv2v< F > &f_)`
Generic implementation of `transform_inplace`.
- `template<typename I , typename F >`
`void transform_inplace (Image< I > &ima_, const Function_v2v< F > &f_)`
Generic implementation of `transform_inplace`.
- `template<typename A , typename I >`
`A::result update (Accumulator< A > &a_, const Image< I > &input_)`
Generic implementation of `data::update`.

9.32.1 Detailed Description

Generic implementation namespace of data namespace.

9.32.2 Function Documentation

9.32.2.1 `template<typename I , typename J > void mln::data::impl::generic::fill_with_image (Image< I > & ima_, const Image< J > & data_)`

Generic implementation.

Parameters

- `[in, out] ima_` The image to be filled.
- `[in] data_` The image.

Definition at line 100 of file `fill_with_image.hh`.

9.32.2.2 `template<typename I , typename V > void mln::data::impl::generic::fill_with_value (Image< I > & ima_, const V & val)`

Fill the whole image *ima* with the single value *v*.

Parameters

- [in, out] *ima_* The image to be filled.
- [in] *val* The value to assign to all sites.

Precondition

ima has to be initialized.

Definition at line 103 of file `fill_with_value.hh`.

9.32.2.3 `template<typename I , typename J > void mln::data::impl::generic::paste (const Image< I > & input_, Image< J > & output_) [inline]`

Generic implementation of [data::paste](#).

Parameters

- [in] *input_* The input image providing pixels values.
- [in, out] *output_* The image in which values are assigned.

Definition at line 111 of file `paste.hh`.

9.32.2.4 `template<typename I , typename F > mln::trait::ch_value< I , typename F ::result >::ret mln::data::impl::generic::transform (const Image< I > & input_, const Function_v2v< F > & f_)`

Generic implementation of [data::transform](#).

Parameters

- [in] *input_* The input image.
- [in] *f_* The function.

Definition at line 137 of file `data/transform.hh`.

References `mln::initialize()`.

9.32.2.5 `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F ::result >::ret mln::data::impl::generic::transform (const Image< I1 > & input1_, const Image< I2 > & input2_, const Function_vv2v< F > & f_)`

Generic implementation of [data::transform](#).

Parameters

- [in] *input1_* The 1st input image.

[in] *input2_* The 2nd input image.

[in] *f_* The function.

Definition at line 166 of file data/transform.hh.

References `mln::initialize()`.

9.32.2.6 `template<typename I1 , typename I2 , typename F > void
mln::data::impl::generic::transform_inplace (Image< I1 > & ima_ , const Image< I2 >
& aux_ , const Function_vv2v< F > & f_)`

Generic implementation of `transform_inplace`.

Parameters

[in] *ima_* The image to be transformed.

[in] *aux_* The auxiliary image.

[in] *f_* The function.

Definition at line 176 of file transform_inplace.hh.

9.32.2.7 `template<typename I , typename F > void mln::data::impl::generic::transform_inplace (
Image< I > & ima_ , const Function_vv2v< F > & f_)`

Generic implementation of `transform_inplace`.

Parameters

[in, out] *ima_* The image to be transformed.

[in] *f_* The function.

Definition at line 149 of file transform_inplace.hh.

9.32.2.8 `template<typename A , typename I > A ::result mln::data::impl::generic::update (
Accumulator< A > & a_ , const Image< I > & input_) [inline]`

Generic implementation of [data::update](#).

Parameters

[in] *a_* The accumulator.

[in] *input_* The input image.

Returns

The accumulator result.

Definition at line 100 of file update.hh.

9.33 mln::data::naive Namespace Reference

Namespace of image processing routines related to pixel levels with naive approach.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::naive](#) namespace.

Functions

- `template<typename I , typename W , typename O >
void median (const Image< I > &input, const Window< W > &win, Image< O > &output)`
Compute in output the median filter of image input by the window win.

9.33.1 Detailed Description

Namespace of image processing routines related to pixel levels with naive approach.

9.33.2 Function Documentation

9.33.2.1 `template<typename I , typename W , typename O > void mln::data::naive::median (
const Image< I > & input, const Window< W > & win, Image< O > & output)
[inline]`

Compute in output the median filter of image input by the window win.

Parameters

- [in] *input* The image to be filtered.
[in] *win* The window.
[in, out] *output* The output image.

This is a NAIVE version for test / comparison purpose so do NOT use it.

Precondition

input and output have to be initialized.

See also

[mln::data::median](#)

Definition at line 99 of file naive/median.hh.

9.34 mln::data::naive::impl Namespace Reference

Implementation namespace of [data::naive](#) namespace.

9.34.1 Detailed Description

Implementation namespace of [data::naive](#) namespace.

9.35 mln::debug Namespace Reference

Namespace of routines that help to debug.

Namespaces

- namespace [impl](#)
Implementation namespace of debug namespace.

Functions

- template<typename I , typename G , typename F >
void [draw_graph](#) ([Image](#)< I > &ima, const [p_vertices](#)< G, F > &pv, typename I::value vcolor, typename I::value ecolord)
Draw an image ima from a mln::p_vertices pv, with value vcolor for vertices, value ecolord for edges and 0 for the background.
- template<typename I , typename G , typename F , typename V , typename E >
void [draw_graph](#) ([Image](#)< I > &ima, const [p_vertices](#)< G, F > &pv, const [Function](#)< V > &vcolor_f_, const [Function](#)< E > &ecolor_f_)
Draw an image ima from a mln::p_vertices pv.
- template<typename I , typename G , typename F , typename V , typename E >
void [draw_graph](#) ([Image](#)< I > &ima, const [p_vertices](#)< [util::line_graph](#)< G >, F > &pv, const [Function](#)< V > &vcolor_f_, const [Function](#)< E > &ecolor_f_)
Draw an image ima from a mln::p_vertices pv.
- std::string [filename](#) (const std::string &filename, int id)
Constructs and returns a formatted output file name.
- signed short [format](#) (signed char v)
Format a signed char to print it properly, i.e., like an integer value.
- unsigned short [format](#) (unsigned char v)
Format an unsigned char to print it properly, i.e., like an integer value.
- template<typename T >
const T & [format](#) (const T &v)
Default version for formatting a value is a no-op.
- char [format](#) (bool v)
Format a Boolean to print it nicely: "|" for true and "-" for false.
- template<typename I >
void [iota](#) ([Image](#)< I > &input, unsigned base_index)
- template<typename I >
mln::trait::concrete< I >::ret [mosaic](#) (const [util::array](#)< I > &input, unsigned n_horizontal, const typename I::value &bg)
Create a single image from an array of image.

- `template<typename I >`
`void println (const std::string &msg, const Image< I > &input)`
Print the message msg and the image input on the standard output.
- `template<typename I >`
`void println (const Image< I > &input)`
Print the image input on the standard output.
- `template<typename I >`
`void println_with_border (const Image< I > &input)`
Print the image input on the standard output.
- `void put_word (image2d< char > &inout, const point2d &word_start, const std::string &word)`
Put the word starting at location word_start in the image inout.
- `template<typename I >`
`image2d< typename I::value > slices_2d (const Image< I > &input, unsigned n_horizontal, unsigned n_vertical, const typename I::value &bg)`
Create a 2D image of the slices of the 3D image input.
- `template<typename I >`
`image2d< typename I::value > slices_2d (const Image< I > &input, float ratio_hv, const typename I::value &bg)`
Create a 2D image of the slices of the 3D image input.
- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &object)`
- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &object_, const value::rgb8 &object_color)`
Superpose two images.
- `template<typename I >`
`void z_order (Image< I > &input)`

9.35.1 Detailed Description

Namespace of routines that help to debug.

9.35.2 Function Documentation

9.35.2.1 `template<typename I , typename G , typename F > void mln::debug::draw_graph (Image< I > & ima, const p_vertices< G, F > & pv, typename I::value vcolor, typename I::value ecolor) [inline]`

Draw an image ima from a [mln::p_vertices](#) pv, with value vcolor for vertices, value ecolor for edges and 0 for the background.

Definition at line 142 of file draw_graph.hh.

References mln::p_vertices< G, F >::graph(), and mln::draw::line().

Referenced by mln::make_debug_graph_image().

9.35.2.2 `template<typename I , typename G , typename F , typename V , typename E > void
mln::debug::draw_graph (Image< I > & ima, const p_vertices< G, F > & pv, const
Function< V > & vcolor_f_, const Function< E > & ecolor_f_) [inline]`

Draw an image *ima* from a mln::p_vertices pv.

Colors for vertices are defined through vcolor_f_. Colors for edges are defined through ecolor_f_.

Definition at line 173 of file draw_graph.hh.

References mln::draw::box_plain(), mln::box< P >::crop_wrt(), mln::p_vertices< G, F >::graph(), and mln::draw::line().

9.35.2.3 `template<typename I , typename G , typename F , typename V , typename E > void
mln::debug::draw_graph (Image< I > & ima, const p_vertices< util::line_graph< G
>, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_)
[inline]`

Draw an image *ima* from a mln::p_vertices pv.

Colors for vertices are defined through vcolor_f_. Colors for edges are defined through ecolor_f_.

Definition at line 211 of file draw_graph.hh.

References mln::p_line2d::begin(), mln::p_line2d::end(), mln::p_vertices< G, F >::graph(), and mln::draw::line().

9.35.2.4 `std::string mln::debug::filename (const std::string & filename, int id = -1)
[inline]`

Constructs and returns a formatted output file name.

The file name is formatted as follow:

‘filename_prefix’_‘id’_‘filename’

Where:

- ‘filename_prefix’ can be set through the global variable debug::internal::filename_prefix.

‘postfix_id’ is autoincremented by default. Its value can be forced.

- ‘filename’ is the given filename

Definition at line 86 of file filename.hh.

9.35.2.5 `signed short mln::debug::format (signed char v) [inline]`

Format a signed char to print it properly, i.e., like an integer value.

Definition at line 78 of file format.hh.

9.35.2.6 unsigned short mln::debug::format (unsigned char v) [inline]

Format an unsigned char to print it properly, i.e., like an integer value.

Definition at line 85 of file format.hh.

9.35.2.7 template<typename T> const T & mln::debug::format (const T & v) [inline]

Default version for formatting a value is a no-op.

Definition at line 64 of file format.hh.

Referenced by mln::value::operator<<(), and mln::Gpoint< E >::operator<<().

9.35.2.8 char mln::debug::format (bool v) [inline]

Format a Boolean to print it nicely: "|" for true and "-" for false.

Definition at line 71 of file format.hh.

9.35.2.9 template<typename I> void mln::debug::iota (Image< I> & input, unsigned base_index) [inline]

Fill the image `input` with successive values.

Parameters

[in, out] *input* The image in which values are assigned.

Definition at line 88 of file debug/iota.hh.

References `iota()`.

Referenced by `iota()`.

9.35.2.10 template<typename I> mln::trait::concrete< I>::ret mln::debug::mosaic (const util::array< I> & input, unsigned n_horizontal, const typename I::value & bg) [inline]

Create a single image from an array of image.

The size of the output image is defined by:

`width = n_horizontal * max(input[i].ncols()) height = (input.size() / n_horizontal) * max(input[i].nrows())`

Returns

a single image where all the input images are displayed as a mosaic.

Definition at line 77 of file mosaic.hh.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

9.35.2.11 `template<typename I> void mln::debug::println (const std::string & msg, const Image< I> & input)`

Print the message `msg` and the image `input` on the standard output.

Definition at line 98 of file `println.hh`.

References `println()`.

9.35.2.12 `template<typename I> void mln::debug::println (const Image< I> & input) [inline]`

Print the image `input` on the standard output.

Definition at line 87 of file `println.hh`.

References `mln::geom::bbox()`.

Referenced by `println()`.

9.35.2.13 `template<typename I> void mln::debug::println_with_border (const Image< I> & input) [inline]`

Print the image `input` on the standard output.

Definition at line 79 of file `println_with_border.hh`.

References `mln::geom::bbox()`.

9.35.2.14 `void mln::debug::put_word (image2d< char> & inout, const point2d & word_start, const std::string & word) [inline]`

Put the `word` starting at location `word_start` in the image `inout`.

Definition at line 55 of file `put_word.hh`.

References `mln::image2d< T>::has()`, and `mln::point< G, C>::last_coord()`.

9.35.2.15 `template<typename I> image2d< typename I::value> mln::debug::slices_2d (const Image< I> & input, unsigned n_horizontal, unsigned n_vertical, const typename I::value & bg) [inline]`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 79 of file `slices_2d.hh`.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

Referenced by `slices_2d()`.

9.35.2.16 `template<typename I> image2d< typename I::value> mln::debug::slices_2d (const Image< I> & input, float ratio_hv, const typename I::value & bg)`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 162 of file `slices_2d.hh`.

References `slices_2d()`.

9.35.2.17 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::debug::superpose (const Image< I > & input, const Image< J > & object)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 106 of file debug/superpose.hh.

References `mln::literal::red`, and `superpose()`.

9.35.2.18 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::debug::superpose (const Image< I > & input_, const Image< J > & object_,
const value::rgb8 & object_color)`

Superpose two images.

Parameters

- [in] *input_* An image. Its value type must be convertible toward `value::rgb8` thanks to a conversion operator or `convert::from_to`.
- [in] *object_* A scalar or labeled image. Objects used for superposition. have their pixel values different from 0.
- [in] *object_color* The color used to draw the objects in *object_*.

Precondition

input_ and *object_* must have the same domain.

Returns

A color image.

Definition at line 79 of file debug/superpose.hh.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

9.35.2.19 `template<typename I > void mln::debug::z_order (Image< I > & input) [inline]`

Fill the image *input* with Z-order (curve) values.

Parameters

- [in, out] *input* The image in which values are assigned.

Reference: [http://en.wikipedia.org/wiki/Z-order_\(curve\)](http://en.wikipedia.org/wiki/Z-order_(curve))

Definition at line 142 of file z_order.hh.

9.36 mln::debug::impl Namespace Reference

Implementation namespace of debug namespace.

9.36.1 Detailed Description

Implementation namespace of debug namespace.

9.37 mln::def Namespace Reference

Namespace for core definitions.

Typedefs

- typedef short [coord](#)
Definition of the default coordinate type: 'short'.
- typedef float [coordf](#)
Definition of the floating coordinate type.

Enumerations

- enum
Definition of the number of bits of the low quantization threshold.

9.37.1 Detailed Description

Namespace for core definitions.

9.37.2 Typedef Documentation

9.37.2.1 typedef short mln::def::coord

Definition of the default coordinate type: 'short'.

Definition at line 43 of file coord.hh.

9.37.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

Definition at line 41 of file coordf.hh.

9.37.3 Enumeration Type Documentation

9.37.3.1 anonymous enum

Definition of the number of bits of the low quantization threshold.

Definition at line 43 of file low_quant_nbits.hh.

9.38 mln::display Namespace Reference

Namespace of routines that help to display images.

Namespaces

- namespace [impl](#)

Implementation namespace of display namespace.

9.38.1 Detailed Description

Namespace of routines that help to display images.

9.39 mln::display::impl Namespace Reference

Implementation namespace of display namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of display namespace.

9.39.1 Detailed Description

Implementation namespace of display namespace.

9.40 mln::display::impl::generic Namespace Reference

Generic implementation namespace of display namespace.

9.40.1 Detailed Description

Generic implementation namespace of display namespace.

9.41 mln::doc Namespace Reference

The namespace [mln::doc](#) is only for documentation purpose.

Classes

- struct [Accumulator](#)
Documentation class for `mln::Accumulator`.
- struct [Box](#)
Documentation class for `mln::Box`.
- struct [Dpoint](#)
Documentation class for `mln::Dpoint`.
- struct [Fastest_Image](#)
Documentation class for the concept of images that have the speed property set to "fastest".
- struct [Generalized_Pixel](#)
Documentation class for `mln::Generalized_Pixel`.
- struct [Image](#)
Documentation class for `mln::Image`.
- struct [Iterator](#)
Documentation class for `mln::Iterator`.
- struct [Neighborhood](#)
Documentation class for `mln::Neighborhood`.
- struct [Object](#)
Documentation class for `mln::Object`.
- struct [Pixel_Iterator](#)
Documentation class for `mln::Iterator`.
- struct [Point_Site](#)
Documentation class for `mln::Point_Site`.
- struct [Site_Iterator](#)
Documentation class for `mln::Site_Iterator`.
- struct [Site_Set](#)
Documentation class for `mln::Site_Set`.
- struct [Value_Iterator](#)
Documentation class for `mln::Value_Iterator`.
- struct [Value_Set](#)
Documentation class for `mln::Value_Set`.
- struct [Weighted_Window](#)
Documentation class for `mln::Weighted_Window`.

- struct [Window](#)

Documentation class for `mln::Window`.

9.41.1 Detailed Description

The namespace `mln::doc` is only for documentation purpose. Since concepts are not yet part of the C++ Standard, they are not explicitly expressed in code. Their documentation is handled by their respective ghost class, located in this namespace.

Warning

The ghost classes located in `mln::doc` should not be used by the client.

9.42 mln::draw Namespace Reference

Namespace of drawing routines.

Functions

- template<typename I , typename B >
void [box](#) ([Image](#)< I > &ima, const [Box](#)< B > &b, const typename I::value &v)
- template<typename I , typename B >
void [box_plain](#) ([Image](#)< I > &ima, const [Box](#)< B > &b, const typename I::value &v)
- template<typename I >
void [dashed_line](#) ([Image](#)< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)
- template<typename I >
void [line](#) ([Image](#)< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename I::value &v)
- template<typename I >
void [plot](#) ([Image](#)< I > &ima, const typename I::point &p, const typename I::value &v)
- template<typename I >
void [polygon](#) ([Image](#)< I > &ima, const [p_array](#)< typename I::site > &par, const typename I::value &v, unsigned output_ratio)
- template<typename I , typename S >
void [site_set](#) ([Image](#)< I > &ima, const [Site_Set](#)< S > &s, const typename I::value &v, unsigned output_ratio=1)

9.42.1 Detailed Description

Namespace of drawing routines.

9.42.2 Function Documentation

- 9.42.2.1** template<typename I , typename B > void mln::draw::box ([Image](#)< I > & *ima*, const [Box](#)< B > & *b*, const typename I::value & *v*) [[inline](#)]

Draw a box at value *v* in image *ima*

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

Precondition

ima has to be initialized.
ima has beg.
ima has end.

Definition at line 72 of file draw/box.hh.

References `line()`.

9.42.2.2 `template<typename I, typename B> void mln::draw::box_plain (Image< I > & ima, const Box< B > & b, const typename I::value & v) [inline]`

Draw a plain box at value *v* in image *ima*

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

Precondition

ima has to be initialized.
ima has beg.
ima has end.

Definition at line 71 of file box_plain.hh.

References `mln::data::fill()`.

Referenced by `mln::debug::draw_graph()`.

9.42.2.3 `template<typename I> void mln::draw::dashed_line (Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v) [inline]`

Draw a dashed line at level *v* in image *ima* between the points *beg* and *end*.

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *beg* The start point to drawn dashed_line.
- [in] *end* The end point to drawn dashed_line.
- [in] *v* The value to assign to all drawn pixels.

Precondition

ima has to be initialized.
ima has *beg*.
ima has *end*.

Definition at line 91 of file `dashed_line.hh`.

References `mln::data::fill()`.

9.42.2.4 `template<typename I> void mln::draw::line (Image< I> & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v)`
[inline]

Draw a line at level *v* in image *ima* between the points *beg* and *end*.

Parameters

[in, out] ***ima*** The image to be drawn.
[in] ***beg*** The start point to drawn line.
[in] ***end*** The end point to drawn line.
[in] ***v*** The value to assign to all drawn pixels.

Precondition

ima has to be initialized.
ima has *beg*.
ima has *end*.

Definition at line 72 of file `draw/line.hh`.

References `mln::data::paste()`.

Referenced by `box()`, `mln::debug::draw_graph()`, and `polygon()`.

9.42.2.5 `template<typename I> void mln::draw::plot (Image< I> & ima, const typename I::point & p, const typename I::value & v)`

Plot a point at level *v* in image *ima*

Parameters

[in, out] ***ima*** The image to be drawn.
[in] ***p*** The point to be plotted.
[in] ***v*** The value to assign to all drawn pixels.

Precondition

ima has to be initialized.
ima has *p*.

9.42.2.6 `template<typename I> void mln::draw::polygon (Image< I > & ima, const p_array< typename I::site > & par, const typename I::value & v, unsigned output_ratio)`

Draw a polygon at level *v* in image *ima*.

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *par* The polygon site set.
- [in] *v* The value to assign to all drawn pixels.

Precondition

ima has to be initialized.

Definition at line 70 of file `polygon.hh`.

References `line()`.

9.42.2.7 `template<typename I, typename S> void mln::draw::site_set (Image< I > & ima, const Site_Set< S > & s, const typename I::value & v, unsigned output_ratio = 1)`

Draw a sites with value *v* in image *ima*

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the site set to draw.
- [in] *v* The value to assign to all drawn pixels.
- [in] *output_ratio* size ratio between output image and the image from which the bboxes were calculated.

Precondition

s is included in *ima* domain.

Definition at line 65 of file `mln/draw/site_set.hh`.

9.43 mln::estim Namespace Reference

Namespace of estimation materials.

Functions

- `template<typename I> mln::value::props< typename I::value >::sum mean (const Image< I > &input)`
Compute the mean value of the pixels of image input.
- `template<typename S, typename I, typename M> void mean (const Image< I > &input, M &result)`

Compute the mean value of the pixels of image `input`.

- `template<typename I >`
`void min_max (const Image< I > &input, typename I::value &min, typename I::value &max)`
Compute the min and max values of the pixels of image `input`.
- `template<typename I >`
`mln::value::props< typename I::value >::sum sum (const Image< I > &input)`
Compute the sum value of the pixels of image `input`.
- `template<typename I , typename S >`
`void sum (const Image< I > &input, S &result)`
Compute the sum value of the pixels of image `input`.

9.43.1 Detailed Description

Namespace of estimation materials.

9.43.2 Function Documentation

9.43.2.1 `template<typename I > mln::value::props< typename I::value >::sum mln::estim::mean (const Image< I > & input) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

[in] *input* The image.

Returns

The mean value.

Definition at line 68 of file `estim/mean.hh`.

References `mln::data::compute()`.

9.43.2.2 `template<typename S , typename I , typename M > void mln::estim::mean (const Image< I > & input, M & result) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

[in] *input* The image.

[out] *result* The mean value.

The free parameter `S` is the type used to compute the summation.

Definition at line 76 of file `estim/mean.hh`.

References `mln::data::compute()`.

9.43.2.3 `template<typename I> void mln::estim::min_max (const Image< I> & input,
typename I::value & min, typename I::value & max) [inline]`

Compute the min and max values of the pixels of image *input*.

Parameters

- [in] *input* The image.
- [out] *min* The minimum pixel value of *input*.
- [out] *max* The maximum pixel value of *input*.

Definition at line 61 of file `estim/min_max.hh`.

References `mln::data::compute()`.

Referenced by `mln::data::impl::stretch()`, and `mln::make::voronoi()`.

9.43.2.4 `template<typename I> mln::value::props< typename I::value>::sum mln::estim::sum
(const Image< I> & input) [inline]`

Compute the sum value of the pixels of image *input*.

Parameters

- [in] *input* The image.

Returns

The sum value.

Definition at line 67 of file `estim/sum.hh`.

References `mln::data::compute()`.

9.43.2.5 `template<typename I, typename S> void mln::estim::sum (const Image< I> & input,
S & result) [inline]`

Compute the sum value of the pixels of image *input*.

Parameters

- [in] *input* The image.
- [out] *result* The sum value.

Definition at line 75 of file `estim/sum.hh`.

References `mln::data::compute()`.

9.44 mln::extension Namespace Reference

Namespace of extension tools.

Functions

- `template<typename I , typename W >`
`void adjust (const Image< I > &ima, const Window< W > &win)`
Adjust the domain extension of image `ima` with the size of the window `win`.
- `template<typename I , typename W >`
`void adjust (const Image< I > &ima, const Weighted_Window< W > &wwin)`
Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.
- `template<typename I >`
`void adjust (const Image< I > &ima, unsigned delta)`
Adjust the domain extension of image `ima` with the size `delta`.
- `template<typename I , typename N >`
`void adjust (const Image< I > &ima, const Neighborhood< N > &nbh)`
Adjust the domain extension of image `ima` with the size of the neighborhood `nbh`.
- `template<typename I , typename W >`
`void adjust_duplicate (const Image< I > &ima, const Window< W > &win)`
Adjust then duplicate.
- `template<typename I , typename W >`
`void adjust_fill (const Image< I > &ima, const Window< W > &win, const typename I::value &val)`
Adjust then fill.
- `template<typename I >`
`void duplicate (const Image< I > &ima)`
Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.
- `template<typename I >`
`void fill (const Image< I > &ima, const typename I::value &val)`

9.44.1 Detailed Description

Namespace of extension tools.

9.44.2 Function Documentation

9.44.2.1 `template<typename I , typename W > void mln::extension::adjust (const Image< I > &ima, const Window< W > & win)`

Adjust the domain extension of image `ima` with the size of the window `win`.

Definition at line 89 of file `extension/adjust.hh`.

References `mln::geom::delta()`.

Referenced by `adjust()`, `adjust_duplicate()`, and `adjust_fill()`.

9.44.2.2 **template<typename I , typename W > void mln::extension::adjust (const Image< I > & *ima*, const Weighted_Window< W > & *wwin*)**

Adjust the domain extension of image *ima* with the size of the weighted window *wwin*.

Definition at line 97 of file extension/adjust.hh.

References adjust(), and mln::geom::delta().

9.44.2.3 **template<typename I > void mln::extension::adjust (const Image< I > & *ima*, unsigned *delta*)**

Adjust the domain extension of image *ima* with the size *delta*.

Definition at line 113 of file extension/adjust.hh.

References adjust().

9.44.2.4 **template<typename I , typename N > void mln::extension::adjust (const Image< I > & *ima*, const Neighborhood< N > & *nbh*)**

Adjust the domain extension of image *ima* with the size of the neighborhood *nbh*.

Definition at line 105 of file extension/adjust.hh.

References adjust(), and mln::geom::delta().

9.44.2.5 **template<typename I , typename W > void mln::extension::adjust_duplicate (const Image< I > & *ima*, const Window< W > & *win*)**

Adjust then duplicate.

Definition at line 70 of file adjust_duplicate.hh.

References adjust(), and duplicate().

9.44.2.6 **template<typename I , typename W > void mln::extension::adjust_fill (const Image< I > & *ima*, const Window< W > & *win*, const typename I::value & *val*)**

Adjust then fill.

Definition at line 72 of file adjust_fill.hh.

References adjust(), and fill().

9.44.2.7 **template<typename I > void mln::extension::duplicate (const Image< I > & *ima*)**

Assign the contents of the domain extension by duplicating the values of the inner boundary of image *ima*.

Definition at line 58 of file extension/duplicate.hh.

Referenced by adjust_duplicate().

9.44.2.8 `template<typename I> void mln::extension::fill (const Image< I> & ima, const typename I::value & val)`

Fill the domain extension of image `ima` with the single value `v`.

Parameters

- [in, out] ***ima*** The image whose domain extension is to be filled.
[in] ***val*** The value to assign.

Precondition

`ima` has to be initialized.

Definition at line 173 of file `extension/fill.hh`.

Referenced by `adjust_fill()`.

9.45 `mln::fun` Namespace Reference

Namespace of functions.

Namespaces

- namespace [access](#)
Namespace for access functions.
- namespace [i2v](#)
Namespace of integer-to-value functions.
- namespace [n2v](#)
Namespace of functions from nil to value.
- namespace [p2b](#)
Namespace of functions from point to boolean.
- namespace [p2p](#)
Namespace of functions from grid point to grid point.
- namespace [p2v](#)
Namespace of functions from point to value.
- namespace [stat](#)
Namespace of statistical functions.
- namespace [v2b](#)
Namespace of functions from value to logic value.
- namespace [v2i](#)
Namespace of value-to-integer functions.

- namespace [v2v](#)
Namespace of functions from value to value.
- namespace [v2w2v](#)
Namespace of bijective functions.
- namespace [v2w_w2v](#)
Namespace of functions from value to value.
- namespace [vv2b](#)
Namespace of functions from value to value.
- namespace [vv2v](#)
Namespace of functions from a couple of values to a value.
- namespace [x2p](#)
Namespace of functions from point to value.
- namespace [x2v](#)
Namespace of functions from vector to value.
- namespace [x2x](#)
Namespace of functions from vector to vector.

Classes

- struct [from_accu](#)
Wrap an accumulator into a function.

9.45.1 Detailed Description

Namespace of functions. Forward declarations.

`fun::i2v::array`

Forward declaration.

9.46 mln::fun::access Namespace Reference

Namespace for access functions.

9.46.1 Detailed Description

Namespace for access functions.

9.47 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

Functions

- `template<typename T >`
`std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`
Operator<<.

9.47.1 Detailed Description

Namespace of integer-to-value functions.

9.47.2 Function Documentation

- 9.47.2.1** `template<typename T > std::ostream & mln::fun::i2v::operator<< (std::ostream & ostr, const array< T > & a)`

Operator<<.

Definition at line 353 of file fun/i2v/array.hh.

9.48 mln::fun::n2v Namespace Reference

Namespace of functions from nil to value.

Classes

- struct [white_gaussian](#)
Generate a White Gaussian Noise.

9.48.1 Detailed Description

Namespace of functions from nil to value.

9.49 mln::fun::p2b Namespace Reference

Namespace of functions from point to boolean.

Classes

- struct [antilogy](#)
A [p2b](#) function always returning `false`.
- struct [tautology](#)
A [p2b](#) function always returning `true`.

9.49.1 Detailed Description

Namespace of functions from point to boolean.

9.50 mln::fun::p2p Namespace Reference

Namespace of functions from grid point to grid point.

9.50.1 Detailed Description

Namespace of functions from grid point to grid point.

9.51 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

9.51.1 Detailed Description

Namespace of functions from point to value.

9.52 mln::fun::stat Namespace Reference

Namespace of statistical functions.

9.52.1 Detailed Description

Namespace of statistical functions.

9.53 mln::fun::v2b Namespace Reference

Namespace of functions from value to logic value.

Classes

- struct [lnot](#)
Functor computing logical-not on a value.
- struct [threshold](#)
Threshold function.

9.53.1 Detailed Description

Namespace of functions from value to logic value.

9.54 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

9.54.1 Detailed Description

Namespace of value-to-integer functions.

9.55 mln::fun::v2v Namespace Reference

Namespace of functions from value to value.

Classes

- class [ch_function_value](#)
Wrap a function [v2v](#) and convert its result to another type.
- struct [component](#)
Functor that accesses the i -th component of a value.
- struct [l1_norm](#)
L1-norm.
- struct [l2_norm](#)
L2-norm.
- struct [linear](#)
*Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.*
- struct [linfty_norm](#)
L-infty norm.

- struct [rgb8_to_rgn](#)
Convert a rgb8 value to a rgn, $n < 8$.

Variables

- [f_hsi_to_rgb_3x8_t](#) [f_hsi_to_rgb_3x8](#)
Global variable.
- [f_hsl_to_rgb_3x8_t](#) [f_hsl_to_rgb_3x8](#)
Global variables.
- [f_rgb_to_hsi_f_t](#) [f_rgb_to_hsi_f](#)
Global variables.
- [f_rgb_to_hsl_f_t](#) [f_rgb_to_hsl_f](#)
Global variables.

9.55.1 Detailed Description

Namespace of functions from value to value.

9.55.2 Variable Documentation

9.55.2.1 [f_hsi_to_rgb_3x8_t](#) mln::fun::v2v::f_hsi_to_rgb_3x8

Global variable.

Definition at line 74 of file hsi_to_rgb.hh.

9.55.2.2 [f_hsl_to_rgb_3x8_t](#) mln::fun::v2v::f_hsl_to_rgb_3x8

Global variables.

Definition at line 92 of file hsl_to_rgb.hh.

9.55.2.3 [f_rgb_to_hsi_f_t](#) mln::fun::v2v::f_rgb_to_hsi_f

Global variables.

Definition at line 66 of file rgb_to_hsi.hh.

9.55.2.4 [f_rgb_to_hsl_f_t](#) mln::fun::v2v::f_rgb_to_hsl_f

Global variables.

Definition at line 75 of file rgb_to_hsl.hh.

9.56 mln::fun::v2w2v Namespace Reference

Namespace of bijective functions.

Classes

- struct [cos](#)
Cosinus bijective functor.

9.56.1 Detailed Description

Namespace of bijective functions.

9.57 mln::fun::v2w_w2v Namespace Reference

Namespace of functions from value to value.

Classes

- struct [l1_norm](#)
L1-norm.
- struct [l2_norm](#)
L2-norm.
- struct [linfty_norm](#)
L-infty norm.

9.57.1 Detailed Description

Namespace of functions from value to value.

9.58 mln::fun::vv2b Namespace Reference

Namespace of functions from value to value.

Classes

- struct [eq](#)
Functor computing equal between two values.
- struct [ge](#)
Functor computing "greater or equal than" between two values.

- struct [gt](#)
Functor computing "greater than" between two values.
- struct [implies](#)
Functor computing logical-implies between two values.
- struct [le](#)
Functor computing "lower or equal than" between two values.
- struct [lt](#)
Functor computing "lower than" between two values.

9.58.1 Detailed Description

Namespace of functions from value to value.

9.59 mln::fun::vv2v Namespace Reference

Namespace of functions from a couple of values to a value.

Classes

- struct [diff_abs](#)
A functor computing the diff_absimum of two values.
- struct [land](#)
Functor computing logical-and between two values.
- struct [land_not](#)
Functor computing logical and-not between two values.
- struct [lor](#)
Functor computing logical-or between two values.
- struct [lxor](#)
Functor computing logical-xor between two values.
- struct [max](#)
A functor computing the maximum of two values.
- struct [min](#)
A functor computing the minimum of two values.
- struct [vec](#)
A functor computing the vecimum of two values.

9.59.1 Detailed Description

Namespace of functions from a couple of values to a value.

9.60 mln::fun::x2p Namespace Reference

Namespace of functions from point to value.

Classes

- struct [closest_point](#)
FIXME: doxygen + concept checking.

9.60.1 Detailed Description

Namespace of functions from point to value.

9.61 mln::fun::x2v Namespace Reference

Namespace of functions from vector to value.

Classes

- struct [bilinear](#)
Represent a bilinear interolation of values from an underlying image.
- struct [trilinear](#)
Represent a trilinear interolation of values from an underlying image.

9.61.1 Detailed Description

Namespace of functions from vector to value.

9.62 mln::fun::x2x Namespace Reference

Namespace of functions from vector to vector.

Classes

- struct [composed](#)
Represent a composition of two transformations.

- struct [linear](#)
Represent a linear interolation of values from an underlying image.
- struct [rotation](#)
Represent a rotation function.
- struct [translation](#)
Translation function-object.

9.62.1 Detailed Description

Namespace of functions from vector to vector.

9.63 mln::geom Namespace Reference

Namespace of all things related to geometry.

Namespaces

- namespace [impl](#)
Implementation namespace of geom namespace.

Classes

- class [complex_geometry](#)
A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

Functions

- `template<typename S >
box< typename S::site > bbox (const Site_Set< S > &pset)`
Compute the precise bounding box of a point set pset.
- `template<typename I >
box< typename I::site > bbox (const Image< I > &ima)`
Compute the precise bounding box of a point set pset.
- `template<typename W >
box< typename W::psite > bbox (const Window< W > &win)`
Compute the precise bounding box of a window win.
- `template<typename W >
box< typename W::psite > bbox (const Weighted_Window< W > &win)`
Compute the precise bounding box of a weighted window win.

- `template<typename I, typename W >`
`mln::trait::ch_value< I, unsigned >::ret chamfer (const Image< I > &input_, const W &w_win_, unsigned max=mln_max(unsigned))`
Apply chamfer algorithm to a binary image.
- `template<typename W >`
`unsigned delta (const Window< W > &win)`
Compute the delta of a window win.
- `template<typename W >`
`unsigned delta (const Weighted_Window< W > &wwin)`
Compute the delta of a weighted window wwin.
- `template<typename N >`
`unsigned delta (const Neighborhood< N > &nbh)`
Compute the delta of a neighborhood nbh.
- `template<typename I >`
`I::site::coord max_col (const Image< I > &ima)`
Give the maximum column of an image.
- `template<typename B >`
`B::site::coord max_col (const Box< B > &b)`
Give the maximum col of an box 2d or 3d.
- `template<typename I >`
`I::site::coord max_ind (const Image< I > &ima)`
Give the maximum ind of an image.
- `template<typename B >`
`B::site::coord max_row (const Box< B > &b)`
Give the maximum row of an box 2d or 3d.
- `template<typename I >`
`I::site::coord max_row (const Image< I > &ima)`
Give the maximum row of an image.
- `template<typename I >`
`I::site::coord max_sli (const Image< I > &ima)`
Give the maximum sli of an image.
- `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec< 3, float > >, complex_image< 2, mln::space_2complex_geometry, float > > mesh_corner_point_area (const p_complex< 2, space_2complex_geometry > &mesh)`
Compute the area “belonging” to normals at vertices.
- `std::pair< complex_image< 2, mln::space_2complex_geometry, float >, complex_image< 2, mln::space_2complex_geometry, float > > mesh_curvature (const p_complex< 2, space_2complex_geometry > &mesh)`
Compute the principal curvatures of a surface at vertices.

- `complex_image< 2, mln::space_2complex_geometry, algebra::vec< 3, float > > mesh_normal`
(const `p_complex< 2, space_2complex_geometry > &mesh`)
Compute normals at vertices.
- `template<typename I >`
`I::site::coord min_col` (const `Image< I > &ima`)
Give the minimum column of an image.
- `template<typename B >`
`B::site::coord min_col` (const `Box< B > &b`)
Give the minimum column of an box 2d or 3d.
- `template<typename I >`
`I::site::coord min_ind` (const `Image< I > &ima`)
Give the minimum ind of an image.
- `template<typename I >`
`I::site::coord min_row` (const `Image< I > &ima`)
Give the minimum row of an image.
- `template<typename B >`
`B::site::coord min_row` (const `Box< B > &b`)
Give the minimum row of an box 2d or 3d.
- `template<typename I >`
`I::site::coord min_sli` (const `Image< I > &ima`)
Give the minimum sli of an image.
- `template<typename I >`
`unsigned ncols` (const `Image< I > &ima`)
Give the number of columns of an image.
- `template<typename B >`
`unsigned ncols` (const `Box< B > &b`)
Give the number of cols of a box 2d or 3d.
- `template<typename I >`
`unsigned ninds` (const `Image< I > &ima`)
Give the number of inds of an image.
- `template<typename I >`
`unsigned nrows` (const `Image< I > &ima`)
Give the number of rows of an image.
- `template<typename B >`
`unsigned nrows` (const `Box< B > &b`)
Give the number of rows of a box 2d or 3d.
- `template<typename I >`
`unsigned nsites` (const `Image< I > &input`)

Compute the number of sites of the image `input`.

- `template<typename I >`
`unsigned nslis (const Image< I > &ima)`
Give the number of slices of an image.
- `template<typename S >`
`void pmin_pmax (const Site_Set< S > &s, typename S::site &pmin, typename S::site &pmax)`
Compute the minimum and maximum points, `pmin` and `max`, of point set `s`.
- `template<typename I >`
`std::pair< typename I::site, typename I::site > pmin_pmax (const Site_Iterator< I > &p)`
Compute the minimum and maximum points when browsing with iterator `p`.
- `template<typename I >`
`void pmin_pmax (const Site_Iterator< I > &p, typename I::site &pmin, typename I::site &pmax)`
Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.
- `template<typename S >`
`std::pair< typename S::site, typename S::site > pmin_pmax (const Site_Set< S > &s)`
Compute the minimum and maximum points of point set `s`.
- `template<typename I , typename Ext >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)`
- `template<typename B >`
`B rotate (const Box< B > &box, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.
- `template<typename B >`
`B rotate (const Box< B > &box_, double angle, const typename B::site &ref)`
Rotate a box.
- `template<typename I >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.
- `template<typename I , typename Ext , typename S >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension, const Site_Set< S > &output_domain)`
Perform a rotation from the center of an image.
- `template<typename I , typename N >`
`mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh)`
Take a labeled image `ima_` with seeds and extend them until creating tiles.
- `template<typename I , typename V >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

- `template<typename I, typename V, typename Ext >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension)`
- `template<typename I, typename V, typename Ext, typename S >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension, const Site_Set< S > &output_domain)`
Perform a translation from the center of an image.
- `template<typename I >`
`mln::trait::concrete< I >::ret vertical_symmetry (const Image< I > &input)`
Performs a vertical symmetry.
- `template<typename I, typename N >`
`I seeds2tiling_roundness (Image< I > &ima_, const w_window2d_int &w_win, unsigned max, const Neighborhood< N > &nbh_)`
Take a labeled image `ima_` with seeds and extend them until creating tiles rounder than the primary version.

9.63.1 Detailed Description

Namespace of all things related to geometry. Namespace of essential things related to geometry.

9.63.2 Function Documentation

9.63.2.1 `template<typename S > box< typename S::site > mln::geom::bbox (const Site_Set< S > & pset) [inline]`

Compute the precise bounding box of a point set `pset`.

Definition at line 122 of file `geom/bbox.hh`.

Referenced by `bbox()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::registration::icp()`, `max_col()`, `max_row()`, `max_sli()`, `min_col()`, `min_row()`, `min_sli()`, `mln::debug::println()`, `mln::debug::println_with_border()`, and `rotate()`.

9.63.2.2 `template<typename I > box< typename I::site > mln::geom::bbox (const Image< I > & ima)`

Compute the precise bounding box of a point set `pset`.

Definition at line 133 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.3 `template<typename W > box< typename W::psite > mln::geom::bbox (const Window< W > & win)`

Compute the precise bounding box of a window `win`.

Definition at line 143 of file `geom/bbox.hh`.

References `mln::literal::origin`.

9.63.2.4 `template<typename W> box< typename W::psite> mln::geom::bbox (const Weighted_Window< W> & win)`

Compute the precise bounding box of a weighted window `win`.

Definition at line 156 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.5 `template<typename I, typename W> mln::trait::ch_value< I, unsigned>::ret mln::geom::chamfer (const Image< I> & input_, const W & w_win_, unsigned max = mln_max(unsigned))`

Apply chamfer algorithm to a binary image.

Definition at line 113 of file `geom/chamfer.hh`.

9.63.2.6 `template<typename W> unsigned mln::geom::delta (const Window< W> & win)`

Compute the delta of a window `win`.

Definition at line 96 of file `delta.hh`.

Referenced by `mln::extension::adjust()`, and `delta()`.

9.63.2.7 `template<typename W> unsigned mln::geom::delta (const Weighted_Window< W> & wwin)`

Compute the delta of a weighted window `wwin`.

Definition at line 105 of file `delta.hh`.

References `delta()`.

9.63.2.8 `template<typename N> unsigned mln::geom::delta (const Neighborhood< N> & nbh)`

Compute the delta of a neighborhood `nbh`.

Definition at line 112 of file `delta.hh`.

References `delta()`.

9.63.2.9 `template<typename I> I::site::coord mln::geom::max_col (const Image< I> & ima) [inline]`

Give the maximum column of an image.

Definition at line 56 of file `max_col.hh`.

References `bbox()`.

Referenced by mln::io::magick::load(), and ncols().

9.63.2.10 `template<typename B > B::site::coord mln::geom::max_col (const Box< B > & b)
[inline]`

Give the maximum col of an box 2d or 3d.

Definition at line 67 of file max_col.hh.

9.63.2.11 `template<typename I > I::site::coord mln::geom::max_ind (const Image< I > & ima
) [inline]`

Give the maximum ind of an image.

Definition at line 51 of file max_ind.hh.

Referenced by ninds().

9.63.2.12 `template<typename B > B::site::coord mln::geom::max_row (const Box< B > & b)
[inline]`

Give the maximum row of an box 2d or 3d.

Definition at line 68 of file max_row.hh.

9.63.2.13 `template<typename I > I::site::coord mln::geom::max_row (const Image< I > & ima
) [inline]`

Give the maximum row of an image.

Definition at line 57 of file max_row.hh.

References bbox().

Referenced by mln::io::magick::load(), and nrows().

9.63.2.14 `template<typename I > I::site::coord mln::geom::max_sli (const Image< I > & ima)
[inline]`

Give the maximum sli of an image.

Definition at line 53 of file max_sli.hh.

References bbox().

Referenced by nslis().

9.63.2.15 `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec<3,
float> >, complex_image< 2, mln::space_2complex_geometry, float > >
mln::geom::mesh_corner_point_area (const p_complex< 2, space_2complex_geometry
> & mesh) [inline]`

Compute the area “belonging” to normals at vertices.

Inspired from the method Trimesh::need_pointareas of the Trimesh library.

See also

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

From the documentation of Trimesh:

“Compute the area "belonging" to each vertex or each corner of a triangle (defined as Voronoi area restricted to the 1-ring of a vertex, or to the triangle).”

Definition at line 249 of file misc.hh.

References `mln::data::fill()`, `mln::norm::sqr_l2()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

Referenced by `mesh_curvature()`.

```
9.63.2.16  std::pair< complex_image< 2, mln::space_2complex_geometry, float
>, complex_image< 2, mln::space_2complex_geometry, float > >
mln::geom::mesh_curvature ( const p_complex< 2, space_2complex_geometry > &
mesh ) [inline]
```

Compute the principal curvatures of a surface at vertices.

These principal curvatures are names `kappa_1` and `kappa_2` in

Sylvie Philipp-Foliguet, Michel Jordan Laurent Najman and Jean Cousty. Artwork 3D Model Database Indexing and Classification.

Parameters

[in] *mesh* The surface (triangle mesh) on which the curvature is to be computed.

Definition at line 486 of file misc.hh.

References `mln::algebra::ldlt_decomp()`, `mln::algebra::ldlt_solve()`, `mesh_corner_point_area()`, `mesh_normal()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

```
9.63.2.17  complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >
mln::geom::mesh_normal ( const p_complex< 2, space_2complex_geometry > & mesh
) [inline]
```

Compute normals at vertices.

Inspired from the method `Trimesh::need_normals` of the Trimesh library.

See also

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

For simplicity purpose, and contrary to Trimesh, this routine only compute normals from a mesh, not from a cloud of points.

Definition at line 161 of file misc.hh.

References `mln::data::fill()`, `mln::norm::sqr_l2()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

Referenced by `mesh_curvature()`.

**9.63.2.18 template<typename I > I::site::coord mln::geom::min_col (const Image< I > & ima)
[inline]**

Give the minimum column of an image.

Definition at line 57 of file min_col.hh.

References bbox().

Referenced by mln::transform::hough(), mln::io::magick::load(), and ncols().

**9.63.2.19 template<typename B > B::site::coord mln::geom::min_col (const Box< B > & b)
[inline]**

Give the minimum column of an box 2d or 3d.

Definition at line 68 of file min_col.hh.

**9.63.2.20 template<typename I > I::site::coord mln::geom::min_ind (const Image< I > & ima
) [inline]**

Give the minimum ind of an image.

Definition at line 51 of file min_ind.hh.

Referenced by ninds().

**9.63.2.21 template<typename I > I::site::coord mln::geom::min_row (const Image< I > & ima
) [inline]**

Give the minimum row of an image.

Definition at line 60 of file min_row.hh.

References bbox().

Referenced by mln::transform::hough(), mln::io::magick::load(), and nrows().

**9.63.2.22 template<typename B > B::site::coord mln::geom::min_row (const Box< B > & b)
[inline]**

Give the minimum row of an box 2d or 3d.

Definition at line 71 of file min_row.hh.

**9.63.2.23 template<typename I > I::site::coord mln::geom::min_sli (const Image< I > & ima)
[inline]**

Give the minimum sli of an image.

Definition at line 53 of file min_sli.hh.

References bbox().

Referenced by nslis().

9.63.2.24 `template<typename I > unsigned mln::geom::ncols (const Image< I > & ima)` `[inline]`

Give the number of columns of an image.

Definition at line 57 of file `ncols.hh`.

References `max_col()`, and `min_col()`.

Referenced by `mln::labeling::impl::compute_fastest()`, `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `ncols()`, `mln::io::magick::save()`, and `mln::subsampling::subsampling()`.

9.63.2.25 `template<typename B > unsigned mln::geom::ncols (const Box< B > & b)`

Give the number of cols of a box 2d or 3d.

Definition at line 67 of file `ncols.hh`.

References `max_col()`, `min_col()`, and `ncols()`.

9.63.2.26 `template<typename I > unsigned mln::geom::ninds (const Image< I > & ima)` `[inline]`

Give the number of inds of an image.

Definition at line 52 of file `ninds.hh`.

References `max_ind()`, and `min_ind()`.

9.63.2.27 `template<typename I > unsigned mln::geom::nrows (const Image< I > & ima)` `[inline]`

Give the number of rows of an image.

Definition at line 57 of file `nrows.hh`.

References `max_row()`, and `min_row()`.

Referenced by `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `nrows()`, `mln::io::magick::save()`, and `mln::subsampling::subsampling()`.

9.63.2.28 `template<typename B > unsigned mln::geom::nrows (const Box< B > & b)`

Give the number of rows of a box 2d or 3d.

Definition at line 66 of file `nrows.hh`.

References `max_row()`, `min_row()`, and `nrows()`.

9.63.2.29 `template<typename I > unsigned mln::geom::nsites (const Image< I > & input)` `[inline]`

Compute the number of sites of the image `input`.

Definition at line 52 of file `nsites.hh`.

Referenced by `pmin_pmax()`.

9.63.2.30 `template<typename I> unsigned mln::geom::nslis (const Image< I> & ima)
[inline]`

Give the number of slices of an image.

Definition at line 53 of file nslis.hh.

References `max_sli()`, and `min_sli()`.

9.63.2.31 `template<typename S> void mln::geom::pmin_pmax (const Site_Set< S> & s,
typename S::site & pmin, typename S::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, of point set `s`.

Definition at line 148 of file pmin_pmax.hh.

References `nsites()`.

9.63.2.32 `template<typename I> std::pair< typename I::site, typename I::site>
mln::geom::pmin_pmax (const Site_Iterator< I> & p) [inline]`

Compute the minimum and maximum points when browsing with iterator `p`.

Definition at line 106 of file pmin_pmax.hh.

References `pmin_pmax()`.

9.63.2.33 `template<typename I> void mln::geom::pmin_pmax (const Site_Iterator< I> & p,
typename I::site & pmin, typename I::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.

Definition at line 84 of file pmin_pmax.hh.

9.63.2.34 `template<typename S> std::pair< typename S::site, typename S::site>
mln::geom::pmin_pmax (const Site_Set< S> & s) [inline]`

Compute the minimum and maximum points of point set `s`.

Definition at line 157 of file pmin_pmax.hh.

References `nsites()`.

Referenced by `pmin_pmax()`.

9.63.2.35 `template<typename B> B mln::geom::rotate (const Box< B> & box, double angle)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.

Definition at line 252 of file rotate.hh.

References `rotate()`.

9.63.2.36 `template<typename I > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & input, double angle)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 197 of file rotate.hh.

References [rotate\(\)](#), and [mln::literal::zero](#).

9.63.2.37 `template<typename B > B mln::geom::rotate (const Box< B > & box_, double angle, const typename B::site & ref)`

Rotate a box.

FIXME: the return type may be too generic and may lead to invalid covariance.

Definition at line 205 of file rotate.hh.

References [mln::compose\(\)](#), [mln::literal::origin](#), and [mln::accu::shape::bbox< P >::to_result\(\)](#).

9.63.2.38 `template<typename I , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension, const Site_Set< S > & output_domain)`

Perform a rotation from the center of an image.

Parameters

[in] *input* An image.

[in] *angle* An angle in degrees.

[in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the rotation.

[in] *output_domain* The domain of the output image. An invalid domain, causes the routine to use a domain large enough to display the whole original image.

Returns

An image with the same domain as *input*.

Definition at line 123 of file rotate.hh.

References [bbox\(\)](#), [mln::compose\(\)](#), [mln::duplicate\(\)](#), [mln::initialize\(\)](#), [mln::mln_exact\(\)](#), [mln::literal::origin](#), and [mln::data::paste\(\)](#).

Referenced by [rotate\(\)](#).

9.63.2.39 `template<typename I , typename Ext > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 185 of file rotate.hh.

References [rotate\(\)](#).

9.63.2.40 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::geom::seeds2tiling (const Image< I > & ima_, const Neighborhood< N > & nbh
) [inline]`

Take a labeled image *ima_* with seeds and extend them until creating tiles.

Parameters

[in, out] *ima_* The labeled image with seed.
[in] *nbh* The neighborhood to use on this algorithm.

Returns

A tiled image.

Precondition

ima_ has to be initialized.

Definition at line 136 of file seeds2tiling.hh.

References `mln::geom::impl::seeds2tiling()`.

9.63.2.41 `template<typename I , typename N > I mln::geom::seeds2tiling_roundness (Image< I
> & ima_, const w_window2d_int & w_win, unsigned max, const Neighborhood< N
> & nbh_) [inline]`

Take a labeled image *ima_* with seeds and extend them until creating tiles rounder than the primary version.

Parameters

[in, out] *ima_* The labeled image with seed.
[in] *w_win* The weight window using by [geom::chamfer](#) to compute distance.
[in] *max* Unsigned using by [geom::chamfer](#) to compute the distance.
[in] *nbh_* The neighborhood to use on this algorithm.

Precondition

ima_ has to be initialized.

Definition at line 128 of file seeds2tiling_roundness.hh.

9.63.2.42 `template<typename I , typename V > mln::trait::concrete< I >::ret
mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V
> & ref)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 146 of file translate.hh.

References `translate()`, and `mln::literal::zero`.

9.63.2.43 `template<typename I , typename V , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension, const Site_Set< S > & output_domain)`

Perform a translation from the center of an image.

Parameters

- [in] *input* An image.
- [in] *ref* The translation vector.
- [in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the translation.
- [in] *output_domain* The domain of the output image. An invalid domain, causes the routine to use the translated *input_domain*.

Returns

An image with the same domain as *input*.

Definition at line 99 of file `translate.hh`.

References `mln::extend()`, `mln::data::fill()`, and `mln::mln_exact()`.

Referenced by `translate()`.

9.63.2.44 `template<typename I , typename V , typename Ext > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 132 of file `translate.hh`.

References `translate()`.

9.63.2.45 `template<typename I > mln::trait::concrete< I >::ret mln::geom::vertical_symmetry (const Image< I > & input)`

Performs a vertical symmetry.

Definition at line 174 of file `vertical_symmetry.hh`.

9.64 mln::geom::impl Namespace Reference

Implementation namespace of `geom` namespace.

Functions

- `template<typename I , typename N > mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh_)`

Generic implementation of `geom::seed2tiling`.

9.64.1 Detailed Description

Implementation namespace of geom namespace.

9.64.2 Function Documentation

9.64.2.1 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::geom::impl::seeds2tiling (const Image< I > & ima_ , const Neighborhood< N > &
nbh_) [inline]`

Generic implementation of geom::seed2tiling.

Parameters

[in, out] *ima_* The labeled image with seed.

[in] *nbh_* The neighborhood to use on this algorithm.

Definition at line 77 of file seeds2tiling.hh.

References `mln::duplicate()`, `mln::p_queue< P >::front()`, `mln::p_queue< P >::pop()`, and `mln::p_queue< P >::push()`.

Referenced by `mln::geom::seeds2tiling()`.

9.65 mln::graph Namespace Reference

Namespace of graph related routines.

Functions

- `template<typename G , typename F >
F::result compute (const Graph< G > &g_ , F &functor)`
Base routine to compute attributes on a graph.
- `template<typename I , typename N , typename L >
mln::trait::ch_value< I , L >::ret labeling (const Image< I > &graph_image_ , const Neighborhood<
N > &nbh_ , L &nlabels)`
Label graph components.
- `template<typename I , typename M >
graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > to_neighb (const Image< I
> &graph_image_ , const Image< M > &graph_mask_image_)`
Make a custom graph neighborhood from a mask image.
- `template<typename I , typename M >
graph_elt_window_if< mln_graph(I), typename I::domain_t, M > to_win (const Image< I >
&graph_image_ , const Image< M > &graph_mask_image_)`
Make a custom graph window from a mask image.

9.65.1 Detailed Description

Namespace of graph related routines.

9.65.2 Function Documentation

9.65.2.1 `template<typename G , typename F > F::result mln::graph::compute (const Graph< G > & g_, F & functor)`

Base routine to compute attributes on a graph.

Parameters

- [in] *g_* A graph.
- [in] *functor* A functor implementing the right interface.

Returns

The computed data.

See also

`canvas::browsing::depth_first_search`

Definition at line 63 of file `graph/compute.hh`.

9.65.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::graph::labeling (const Image< I > & graph_image_, const Neighborhood< N > & nbh_, L & nlabels)`

Label graph components.

[Vertex](#) with id 0, usually used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

Parameters

- [in] *graph_image_* A graph image (

See also

[vertex_image](#), [edge_image](#)).

Parameters

- [in] *nbh_* A graph neighborhood.
- [in, out] *nlabels* The number of labels found.

Returns

a [Graph](#) image of labels.

Definition at line 72 of file `labeling.hh`.

References `mln::labeling::blobs()`, `mln::data::fill()`, and `mln::initialize()`.

9.65.2.3 `template<typename I , typename M > graph_elt_neighborhood_if< mln_graph(I),
typename I::domain_t, M > mln::graph::to_neighb (const Image< I > & graph_image_,
const Image< M > & graph_mask_image_)`

Make a custom graph neighborhood from a mask image.

Parameters

[in] *graph_image_* A graph image (

See also

[vertex_image](#) and [edge_image](#)).

Parameters

[in] *graph_mask_image_* A graph image of bool used as a mask.

Returns

A masked neighborhood on graph.

Definition at line 57 of file to_neighb.hh.

9.65.2.4 `template<typename I , typename M > graph_elt_window_if< mln_graph(I), typename
I::domain_t, M > mln::graph::to_win (const Image< I > & graph_image_, const
Image< M > & graph_mask_image_)`

Make a custom graph window from a mask image.

Parameters

[in] *graph_image_* A graph image (

See also

[vertex_image](#) and [edge_image](#)).

Parameters

[in] *graph_mask_image_* A graph image of bool used as a mask.

Returns

A masked window on graph.

Definition at line 57 of file to_win.hh.

9.66 mln::grid Namespace Reference

Namespace of grids definitions.

9.66.1 Detailed Description

Namespace of grids definitions. Compute the image::space trait from a point type.

9.67 mln::histo Namespace Reference

Namespace of histograms.

Namespaces

- namespace [impl](#)
Implementation namespace of histo namespace.

Classes

- struct [array](#)
Generic histogram class over a value set with type T.

Functions

- `template<typename I >`
`histo::array< typename I::value > compute (const Image< I > &input)`
Compute the histogram of image input.
- `template<typename I >`
`mln::trait::concrete< I >::ret equalize (const Image< I > &input)`
Equalizes the histogram of image input.

9.67.1 Detailed Description

Namespace of histograms.

9.67.2 Function Documentation

9.67.2.1 `template<typename I > histo::array< typename I::value > mln::histo::compute (const Image< I > & input) [inline]`

Compute the histogram of image `input`.

Definition at line 79 of file `histo/compute.hh`.

Referenced by `equalize()`.

9.67.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::histo::equalize (const Image< I > & input)`

Equalizes the histogram of image `input`.

Author

J. Fabrizio, R. Levillain

Definition at line 55 of file histo/equalize.hh.

References `compute()`, and `mln::initialize()`.

9.68 mln::histo::impl Namespace Reference

Implementation namespace of histo namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of histo namespace.

9.68.1 Detailed Description

Implementation namespace of histo namespace.

9.69 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of histo namespace.

9.69.1 Detailed Description

Generic implementation namespace of histo namespace.

9.70 mln::impl Namespace Reference

Implementation namespace of mln namespace.

9.70.1 Detailed Description

Implementation namespace of mln namespace.

9.71 mln::io Namespace Reference

Namespace of input/output handling.

Namespaces

- namespace [cloud](#)
Namespace of cloud input/output handling.

- namespace [dicom](#)
Namespace of DICOM input/output handling.
- namespace [dump](#)
Namespace of dump input/output handling.
- namespace [fits](#)
Namespace of fits input/output handling.
- namespace [fld](#)
Namespace of pgm input/output handling.
- namespace [magick](#)
Namespace of magick input/output handling.
- namespace [off](#)
Namespace of off input/output handling.
- namespace [pbm](#)
Namespace of pbm input/output handling.
- namespace [pbms](#)
Namespace of pbms input/output handling.
- namespace [pfm](#)
Namespace of pfm input/output handling.
- namespace [pgm](#)
Namespace of pgm input/output handling.
- namespace [pgms](#)
Namespace of pgms input/output handling.
- namespace [plot](#)
Namespace of plot input/output handling.
- namespace [pnm](#)
Namespace of pnm input/output handling.
- namespace [pnms](#)
Namespace of pnms input/output handling.
- namespace [ppm](#)
Namespace of ppm input/output handling.
- namespace [ppms](#)
Namespace of ppms input/output handling.
- namespace [raw](#)
Namespace of raw input/output handling.

- namespace [tiff](#)
Namespace of tiff input/output handling.
- namespace [txt](#)
Namespace of txt input/output handling.

9.71.1 Detailed Description

Namespace of input/output handling.

9.72 mln::io::cloud Namespace Reference

Namespace of cloud input/output handling.

Functions

- `template<typename P >
void load (p_array< P > &arr, const std::string &filename)`
Load a cloud of points.
- `template<typename P >
void save (const p_array< P > &arr, const std::string &filename)`
Load a cloud of points.

9.72.1 Detailed Description

Namespace of cloud input/output handling.

9.72.2 Function Documentation

9.72.2.1 `template<typename P > void mln::io::cloud::load (p_array< P > & arr, const std::string & filename)`

Load a cloud of points.

Parameters

- [in, out] **arr** the site set where to load the data.
[in] **filename** file to load.

Definition at line 88 of file cloud/load.hh.

9.72.2.2 `template<typename P > void mln::io::cloud::save (const p_array< P > & arr, const std::string & filename)`

Load a cloud of points.

Parameters

- [in] *arr* the cloud of points to save.
- [in] *filename* the destination.

Definition at line 83 of file cloud/save.hh.

9.73 `mln::io::dicom` Namespace Reference

Namespace of DICOM input/output handling.

Classes

- struct `dicom_header`
Store dicom file header.

Functions

- `dicom_header get_header` (const std::string &filename)
Retrieve header in a dicom file.
- `template<typename I > void load (Image< I > &ima, const std::string &filename)`

9.73.1 Detailed Description

Namespace of DICOM input/output handling.

9.73.2 Function Documentation

9.73.2.1 `dicom_header mln::io::dicom::get_header (const std::string & filename)`

Retrieve header in a dicom file.

Definition at line 76 of file dicom/get_header.hh.

References `mln::util::array< T >::append()`.

9.73.2.2 `template<typename I > void mln::io::dicom::load (Image< I > & ima, const std::string & filename) [inline]`

Load a DICOM file in a Milena image.

Parameters

- [out] *ima* A reference to the image which will receive data.
- [in] *filename* The source.

Common compilation flags to link to gdcm if this file is used:

-lgdcmCommon -lgdcmDICT -lgdcmDSED -lgdcmIOD -lgdcmMSFF -lgdcmexpat -lgdcmjpeg12 -lgdcmjpeg16 -lgdcmjpeg8 -lgdcmopenjpeg -lgdcmuuid -lgdcmzlib

Definition at line 96 of file dicom/load.hh.

References mln::initialize().

9.74 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

Classes

- struct [dump_header](#)
Store dump file header.

Functions

- [dump_header get_header](#) (const std::string &filename)
Retrieve header in a dump file.
- template<typename I >
void [load](#) ([Image](#)< I > &ima_, const std::string &filename)
Load a Milena image by dumped into a file.
- template<typename I >
void [save](#) (const [Image](#)< I > &ima_, const std::string &filename)
Save a Milena image by dumping its data to a file.

9.74.1 Detailed Description

Namespace of dump input/output handling.

9.74.2 Function Documentation**9.74.2.1 dump_header mln::io::dump::get_header (const std::string & filename)**

Retrieve header in a dump file.

Definition at line 68 of file dump/get_header.hh.

References mln::util::array< T >::resize().

9.74.2.2 `template<typename I> void mln::io::dump::load (Image< I> & ima_, const std::string & filename)`

Load a Milena image by dumped into a file.

Parameters

[in, out] *ima_* The image to load.

[in] *filename* the destination.

Definition at line 171 of file dump/load.hh.

9.74.2.3 `template<typename I> void mln::io::dump::save (const Image< I> & ima_, const std::string & filename)`

Save a Milena image by dumping its data to a file.

Parameters

[in] *ima_* The image to save.

[in] *filename* the destination.

Definition at line 131 of file dump/save.hh.

9.75 `mln::io::fits` Namespace Reference

Namespace of fits input/output handling.

Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)
Load a fits image in a Milena image.
- [image2d](#)< float > [load](#) (const std::string &filename)
Load a fits image in a image2d<float>.

9.75.1 Detailed Description

Namespace of fits input/output handling.

9.75.2 Function Documentation

9.75.2.1 `void mln::io::fits::load (image2d< float> & ima, const std::string & filename)` `[inline]`

Load a fits image in a Milena image.

Parameters

[out] *ima* A reference to the image2d<float> which will receive data.

[in] *filename* The source.

Definition at line 132 of file fits/load.hh.

9.75.2.2 image2d<float> mln::io::fits::load (const std::string & filename) [inline]

Load a fits image in a image2d<float>.

Parameters

[in] *filename* The image source.

Returns

An image2d<float> which contains loaded data.

Definition at line 85 of file fits/load.hh.

9.76 mln::io::fld Namespace Reference

Namespace of pgm input/output handling.

Classes

- struct [fld_header](#)
Define the header structure of an AVS field data file.

Functions

- template<typename I>
void [load](#) (Image< I > &ima_, const char *filename)
Load an image from an AVS field file.
- [fld_header read_header](#) (std::istream &ins)
Read the header form an AVS field file.
- void [write_header](#) (std::ostream &file, const [fld_header](#) &h)
Write the AVS header in a file.

9.76.1 Detailed Description

Namespace of pgm input/output handling.

9.76.2 Function Documentation

9.76.2.1 `template<typename I> void mln::io::fld::load (Image< I> & ima_, const char * filename) [inline]`

Load an image from an AVS field file.

Parameters

- `[in, out] ima_` The image to load.
- `[in] filename` The path to the AVS file.

Definition at line 198 of file fld/load.hh.

References `mln::box< P>::pmax()`, `mln::box< P>::pmin()`, and `read_header()`.

9.76.2.2 `fld_header mln::io::fld::read_header (std::istream & ins) [inline]`

Read the header form an AVS field file.

Parameters

- `ins` The file to read.

Returns

- The header.

Definition at line 73 of file fld/load_header.hh.

Referenced by `load()`.

9.76.2.3 `void mln::io::fld::write_header (std::ostream & file, const fld_header & h) [inline]`

Write the AVS header in a file.

Parameters

- `file` The file to write.
- `h` The AVS header.

Definition at line 58 of file write_header.hh.

9.77 mln::io::magick Namespace Reference

Namespace of magick input/output handling.

Functions

- `template<typename I>`
`void load (Image< I> &ima, const std::string &filename)`

Load data from a file into a Milena image using Magick++.

- `template<typename I, typename J >`
`void save (const Image< I > &ima, const Image< J > &opacity_mask, const std::string &filename)`

Save a Milena image into a file using Magick++.

- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

Save a Milena image into a file using Magick++.

9.77.1 Detailed Description

Namespace of magick input/output handling.

9.77.2 Function Documentation

9.77.2.1 `template<typename I > void mln::io::magick::load (Image< I > & ima, const std::string & filename) [inline]`

Load data from a file into a Milena image using Magick++.

Parameters

[out] *ima* The image data are loaded into.

[in] *filename* The name of the input file.

Definition at line 139 of file magick/load.hh.

References `mln::initialize()`, `mln::geom::max_col()`, `mln::geom::max_row()`, `mln::geom::min_col()`, and `mln::geom::min_row()`.

9.77.2.2 `template<typename I, typename J > void mln::io::magick::save (const Image< I > & ima, const Image< J > & opacity_mask, const std::string & filename)`

Save a Milena image into a file using Magick++.

Parameters

[in] *ima* The image to save.

[in] *opacity_mask* Mask used to set pixel opacity_mask in output image. Output format must support this feature to be taken into account.

[in] *filename* The name of the output file.

Definition at line 228 of file magick/save.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.77.2.3 `template<typename I> void mln::io::magick::save (const Image< I> & ima, const std::string & filename) [inline]`

Save a Milena image into a file using Magick++.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *ima* The image to save.
- [in] *filename* The name of the output file.

Definition at line 288 of file magick/save.hh.

9.78 `mln::io::off` Namespace Reference

Namespace of off input/output handling.

Functions

- void `load (bin_2complex_image3df &ima, const std::string &filename)`
Load a (binary) OFF image into a complex image.
- void `save (const bin_2complex_image3df &ima, const std::string &filename)`
Save a (binary) OFF image into a complex image.
- `template<typename I>`
void `save_bin_alt (const I &ima, const std::string &filename)`
FIXME: Similar to `mln::io::off::save(const bin_2complex_image3df&, const std::string&)`, but does not save faces whose value is 'false'.

9.78.1 Detailed Description

Namespace of off input/output handling.

9.78.2 Function Documentation

9.78.2.1 `void mln::io::off::load (bin_2complex_image3df & ima, const std::string & filename)`

Load a (binary) OFF image into a complex image.

Load a 3x8-bit RGB (color) OFF image into a complex image.

Load a floating-point OFF image into a complex image.

Parameters

- [out] *ima* A reference to the image to construct.
- [in] *filename* The name of the file to load.

The image is said binary since data only represent the existence of faces.

Parameters

[out] *ima* A reference to the image to construct.

[in] *filename* The name of the file to load.

Read floating-point data is attached to 2-faces only; 1-faces and 0-faces are set to 0.0f.

Definition at line 195 of file off/load.hh.

9.78.2.2 void mln::io::off::save (const bin_2complex_image3df & *ima*, const std::string & *filename*)

Save a (binary) OFF image into a complex image.

Save a 3x8-bit RGB (color) OFF image into a complex image.

Save a floating-point value grey-level OFF image into a complex image.

Save an 8-bit grey-level OFF image into a complex image.

Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

The image is said binary since data represent only the existence of faces.

Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

Only data is attached to 2-faces is saved; the OFF file cannot store data attached to faces of other dimensions.

Definition at line 189 of file off/save.hh.

9.78.2.3 template<typename I> void mln::io::off::save_bin_alt (const I & *ima*, const std::string & *filename*)

FIXME: Similar to [mln::io::off::save\(const bin_2complex_image3df&, const std::string&\)](#), but does not save faces whose value is 'false'.

Definition at line 59 of file save_bin_alt.hh.

9.79 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbm implementation details.

Functions

- void [load](#) ([image2d](#)< bool > &ima, const std::string &filename)
Load a pbm image in a Milena image.
- [image2d](#)< bool > [load](#) (const std::string &filename)
Load a pbm image in a image2d<float>.
- template<typename I >
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

9.79.1 Detailed Description

Namespace of pbm input/output handling.

9.79.2 Function Documentation

9.79.2.1 void [mln::io::pbm::load](#) ([image2d](#)< bool > & *ima*, const std::string & *filename*)
[inline]

Load a pbm image in a Milena image.

Parameters

- [out] *ima* A reference to the [image2d](#)<bool> which will receive data.
[in] *filename* The source.

Definition at line 156 of file [pbm/load.hh](#).

9.79.2.2 [image2d](#)< bool > [mln::io::pbm::load](#) (const std::string & *filename*) [inline]

Load a pbm image in a [image2d](#)<float>.

Parameters

- [in] *filename* The image source.

Returns

An [image2d](#)<float> which contains loaded data.

Definition at line 128 of file [pbm/load.hh](#).

9.79.2.3 `template<typename I> void mln::io::pbm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a pbm image.

Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Definition at line 118 of file pbm/save.hh.

9.80 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

9.80.1 Detailed Description

Namespace of pbm implementation details.

9.81 mln::io::pbms Namespace Reference

Namespace of pbms input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbms implementation details.

Functions

- void [load](#) ([image3d](#)< bool > &*ima*, const [util::array](#)< std::string > &*filenames*)
Load pbms images as slices of a 3D Milena image.

9.81.1 Detailed Description

Namespace of pbms input/output handling.

9.81.2 Function Documentation

9.81.2.1 `void mln::io::pbms::load (image3d< bool > & ima, const util::array< std::string > & filenames) [inline]`

Load pbms images as slices of a 3D Milena image.

Parameters

[out] *ima* A reference to the 3D image which will receive data.

[in] *filenames* The list of 2D images to load..

Definition at line 65 of file pbms/load.hh.

9.82 mln::io::pbms::impl Namespace Reference

Namespace of pbms implementation details.

9.82.1 Detailed Description

Namespace of pbms implementation details.

9.83 mln::io::pfm Namespace Reference

Namespace of pfm input/output handling.

Namespaces

- namespace [impl](#)

Implementation namespace of pfm namespace.

Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)

Load a pfm image in a Milena image.

- [image2d](#)< float > [load](#) (const std::string &filename)

Load a pfm image in a image2d<float>.

- template<typename I >

void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

Save a Milena image as a pfm image.

9.83.1 Detailed Description

Namespace of pfm input/output handling.

9.83.2 Function Documentation

9.83.2.1 `void mln::io::pfm::load (image2d< float > & ima, const std::string & filename)`
`[inline]`

Load a pfm image in a Milena image.

Parameters

[out] *ima* A reference to the image2d<float> which will receive data.

[in] *filename* The source.

Definition at line 162 of file pfm/load.hh.

9.83.2.2 `image2d< float > mln::io::pfm::load (const std::string & filename)` `[inline]`

Load a pfm image in a image2d<float>.

Parameters

[in] *filename* The image source.

Returns

An image2d<float> which contains loaded data.

Definition at line 138 of file pfm/load.hh.

9.83.2.3 `template<typename I> void mln::io::pfm::save (const Image< I > & ima, const std::string & filename)` `[inline]`

Save a Milena image as a pfm image.

Parameters

[in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 101 of file pfm/save.hh.

9.84 mln::io::pfm::impl Namespace Reference

Implementation namespace of pfm namespace.

9.84.1 Detailed Description

Implementation namespace of pfm namespace.

9.85 mln::io::pgm Namespace Reference

Namespace of pgm input/output handling.

Functions

- `template<typename I >`
`void load (Image< I > &ima, const std::string &filename)`
Load a pgm image in a Milena image.
- `template<typename V >`
`image2d< V > load (const std::string &filename)`
Load a pgm image in a Milena image.
- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

9.85.1 Detailed Description

Namespace of pgm input/output handling.

9.85.2 Function Documentation

9.85.2.1 `template<typename I > void mln::io::pgm::load (Image< I > & ima, const std::string & filename) [inline]`

Load a pgm image in a Milena image.

Parameters

- [out] *ima* A reference to the image which will receive data.
 [in] *filename* The source.

Definition at line 87 of file `pgm/load.hh`.

9.85.2.2 `template<typename V > image2d< V > mln::io::pgm::load (const std::string & filename) [inline]`

Load a pgm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

Parameters

- [in] *filename* The image source.

Returns

An `image2d` which contains loaded data.

Definition at line 77 of file `pgm/load.hh`.

9.85.2.3 `template<typename I > void mln::io::pgm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a pgm image.

Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Definition at line 77 of file pgm/save.hh.

9.86 mln::io::pgms Namespace Reference

Namespace of pgms input/output handling.

Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load pgm images as slices of a 3D Milena image.

9.86.1 Detailed Description

Namespace of pgms input/output handling.

9.86.2 Function Documentation

9.86.2.1 `template<typename V > void mln::io::pgms::load (image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load pgm images as slices of a 3D Milena image.

Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 69 of file pgms/load.hh.

9.87 mln::io::plot Namespace Reference

Namespace of plot input/output handling.

Functions

- `template<typename I >`
`void load (util::array< I > &arr, const std::string &filename)`
- `template<typename T >`
`void save (const histo::array< T > &arr, const std::string &filename)`
- `template<typename T >`
`void save (const util::array< T > &arr, const std::string &filename, int start_value=0)`
Save a Milena array in a plot file.
- `template<typename I >`
`void save (const image1d< I > &ima, const std::string &filename)`
Save a Milena 1D image in a plot file.

9.87.1 Detailed Description

Namespace of plot input/output handling.

9.87.2 Function Documentation

9.87.2.1 `template<typename I > void mln::io::plot::load (util::array< I > & arr, const std::string & filename) [inline]`

Load a Milena 1D image from a plot file.

Parameters

- [in] *ima* A reference to the image to load.
 [out] *filename* The output file.
 [in] *start_value* The start index value of the plot (optional).

Load a Milena array from a plot file.

Parameters

- [in] *arr* A reference to the array to load.
 [out] *filename* The output file.

Definition at line 93 of file plot/load.hh.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::clear()`.

9.87.2.2 `template<typename T > void mln::io::plot::save (const histo::array< T > & arr, const std::string & filename) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 103 of file plot/save.hh.

9.87.2.3 `template<typename T> void mln::io::plot::save (const util::array< T> & arr, const std::string & filename, int start_value = 0) [inline]`

Save a Milena array in a plot file.

Parameters

- [in] *arr* A reference to the array to save.
- [out] *filename* The output file.
- [in] *start_value* The start index value of the plot (optional).

Definition at line 89 of file plot/save.hh.

References `mln::util::array< T>::nelements()`.

9.87.2.4 `template<typename I> void mln::io::plot::save (const image1d< I> & ima, const std::string & filename)`

Save a Milena 1D image in a plot file.

Parameters

- [in] *ima* A reference to the image to save.
- [out] *filename* The output file.

9.88 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pnm's implementation details.

Functions

- `template<typename V> image2d< V> load (char type_, const std::string &filename)`
main function : load pnm format
- `template<typename I> void load (char type_, Image< I> &ima_, const std::string &filename)`
An other way to load pnm files : the destination is an argument to check if the type match the file to load.
- `template<typename I> void load_raw_2d (std::ifstream &file, I &ima)`
load_raw_2d.

- `template<typename V >`
`unsigned int max_component (const V &)`
Give the maximum value which can be stored as a component value type V.
- `template<typename I >`
`void save (char type, const Image< I > &ima_, const std::string &filename)`

9.88.1 Detailed Description

Namespace of pnm input/output handling.

9.88.2 Function Documentation

9.88.2.1 `template<typename V > image2d<V> mln::io::pnm::load (char type_, const std::string & filename) [inline]`

main function : load pnm format

Definition at line 210 of file `pnm/load.hh`.

References `load_raw_2d()`, and `max_component()`.

9.88.2.2 `template<typename I > void mln::io::pnm::load (char type_, Image< I > & ima_, const std::string & filename) [inline]`

An other way to load pnm files : the destination is an argument to check if the type match the file to load.

Definition at line 257 of file `pnm/load.hh`.

References `mln::make::box2d()`, `load_raw_2d()`, and `max_component()`.

9.88.2.3 `template<typename I > void mln::io::pnm::load_raw_2d (std::ifstream & file, I & ima) [inline]`

`load_raw_2d`.

for all pnm 8/16 bits formats

Definition at line 198 of file `pnm/load.hh`.

Referenced by `load()`.

9.88.2.4 `template<typename V > unsigned int mln::io::pnm::max_component (const V &) [inline]`

Give the maximum value which can be stored as a component value type V.

Definition at line 56 of file `max_component.hh`.

Referenced by `load()`.

9.88.2.5 `template<typename I> void mln::io::pnm::save (char type, const Image< I > & ima_, const std::string & filename) [inline]`

Save a Milena image as a pnm image.

Parameters

- [in] *type* The type of the image to save (can be PPM, PGM, PBM).
- [in] *ima_* The image to save.
- [in, out] *filename* the destination.

Definition at line 185 of file pnm/save.hh.

9.89 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

9.89.1 Detailed Description

Namespace of pnm's implementation details.

9.90 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

Functions

- `template<typename V>`
`void load (char type, image3d< V > &ima, const util::array< std::string > &filenames)`
Load pnm images as slices of a 3D Milena image.
- `void load (char type, image3d< bool > &ima, const util::array< std::string > &filenames)`

9.90.1 Detailed Description

Namespace of pnms input/output handling.

9.90.2 Function Documentation

9.90.2.1 `template<typename V> void mln::io::pnms::load (char type, image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load pnm images as slices of a 3D Milena image.

Parameters

- [in] *type* The type of the pnm files.

[out] *ima* A reference to the 3D image which will receive data.

[in] *filenames* The list of 2D images to load..

Definition at line 79 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::nelements()`.

Referenced by `load()`.

9.90.2.2 `void mln::io::pnms::load (char type, image3d< bool > & ima, const util::array< std::string > & filenames) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 102 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `load()`, and `mln::util::array< T >::nelements()`.

9.91 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

Functions

- `template<typename I >`
`void load (Image< I > &ima, const std::string &filename)`
Load a ppm image in a Milena image.
- `template<typename V >`
`image2d< V > load (const std::string &filename)`
Load a ppm image in a Milena image.
- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

9.91.1 Detailed Description

Namespace of ppm input/output handling.

9.91.2 Function Documentation

9.91.2.1 `template<typename I > void mln::io::ppm::load (Image< I > & ima, const std::string & filename) [inline]`

Load a ppm image in a Milena image.

Parameters

[out] *ima* A reference to the image which will receive data.

[in] *filename* The source.

Definition at line 89 of file ppm/load.hh.

9.91.2.2 `template<typename V > image2d< V > mln::io::ppm::load (const std::string & filename) [inline]`

Load a ppm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

Parameters

[in] *filename* The image source.

Returns

An `image2d` which contains loaded data.

Definition at line 79 of file ppm/load.hh.

9.91.2.3 `template<typename I > void mln::io::ppm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a ppm image.

Parameters

[in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 65 of file ppm/save.hh.

Referenced by `mln::registration::icp()`.

9.92 mln::io::ppms Namespace Reference

Namespace of ppms input/output handling.

Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load ppm images as slices of a 3D Milena image.

9.92.1 Detailed Description

Namespace of ppms input/output handling.

9.92.2 Function Documentation

9.92.2.1 `template<typename V > void mln::io::ppms::load (image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load ppm images as slices of a 3D Milena image.

Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 67 of file ppms/load.hh.

9.93 mln::io::raw Namespace Reference

Namespace of raw input/output handling.

Classes

- struct [raw_header](#)
Store raw file header.

Functions

- [raw_header get_header](#) (const std::string &filename)
Retrieve header in a raw file.
- `template<typename I >
void load (Image< I > &ima_, const std::string &filename)`
Load an image saved as a raw data file.
- `template<typename I >
void save (const Image< I > &ima_, const std::string &filename)`
Save a Milena image as a raw data file.

9.93.1 Detailed Description

Namespace of raw input/output handling.

9.93.2 Function Documentation

9.93.2.1 raw_header mln::io::raw::get_header (const std::string & *filename*)

Retrieve header in a raw file.

Definition at line 68 of file raw/get_header.hh.

References mln::util::array< T >::resize().

9.93.2.2 template<typename I > void mln::io::raw::load (Image< I > & *ima_*, const std::string & *filename*)

Load an image saved as a raw data file.

Parameters

[in, out] *ima_* The image to load.

[in] *filename* the destination.

This routine try to read two input files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 184 of file raw/load.hh.

9.93.2.3 template<typename I > void mln::io::raw::save (const Image< I > & *ima_*, const std::string & *filename*)

Save a Milena image as a raw data file.

Parameters

[in] *ima_* The image to save.

[in] *filename* the destination.

This routine produce two output files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 135 of file raw/save.hh.

9.94 mln::io::tiff Namespace Reference

Namespace of tiff input/output handling.

Functions

- template<typename I >
void [load](#) (Image< I > &ima_, const std::string &filename)
Load a TIFF image to a Milena image.

9.94.1 Detailed Description

Namespace of tiff input/output handling.

9.94.2 Function Documentation

9.94.2.1 `template<typename I> void mln::io::tiff::load (Image< I> & ima, const std::string & filename) [inline]`

Load a TIFF image to a Milena image.

Definition at line 323 of file tiff/load.hh.

9.95 mln::io::txt Namespace Reference

Namespace of txt input/output handling.

Functions

- void `save` (const `image2d`< char > &*ima*, const std::string &*filename*)
Save an image as txt file.

9.95.1 Detailed Description

Namespace of txt input/output handling.

9.95.2 Function Documentation

9.95.2.1 `void mln::io::txt::save (const image2d< char > & ima, const std::string & filename) [inline]`

Save an image as txt file.

Parameters

- [in] *ima* The image to save. Must be an image of char.
[in] *filename* the destination.

Definition at line 63 of file txt/save.hh.

References `mln::image2d< T>::domain()`.

9.96 mln::labeling Namespace Reference

Namespace of labeling routines.

Namespaces

- namespace [impl](#)

Implementation namespace of labeling namespace.

Functions

- `template<typename I , typename N , typename L >
mln::trait::ch_value< I, L >::ret background (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I , typename N , typename L >
mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`

Connected component labeling of the binary objects of a binary image.

- `template<typename I , typename N , typename L , typename A >
util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > blobs_and_compute (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)`
- `template<typename V , typename L >
mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image, const typename L::value &nlabels)`

Create a new color image from a labeled image and fill each component with a random color.

- `template<typename V , typename L >
mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image)`
- `template<typename L >
mln::trait::ch_value< L, mln::value::rgb8 >::ret colorize (const Image< L > &input, const typename L::value &nlabels)`
- `template<typename A , typename I , typename L >
util::array< mln_meta_accu_result(A, typename I::value)> compute (const Meta_Accumulator< A > &a, const Image< I > &input, const Image< L > &label, const typename L::value &nlabels)`

Compute an accumulator onto the pixel values of the image input.

- `template<typename A , typename L >
util::array< typename A::result > compute (const Accumulator< A > &a, const Image< L > &label, const typename L::value &nlabels)`

Compute an accumulator onto the pixel sites of each component domain of label.

- `template<typename A , typename L >
util::array< mln_meta_accu_result(A, typename L::psite)> compute (const Meta_Accumulator< A > &a, const Image< L > &label, const typename L::value &nlabels)`

Compute an accumulator onto the pixel sites of each component domain of label.

- `template<typename A , typename I , typename L >
util::array< typename A::result > compute (util::array< A > &a, const Image< I > &input, const Image< L > &label, const typename L::value &nlabels)`

Compute an accumulator onto the pixel values of the image input.

- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< I > &input, const Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image input.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const util::array< typename A::result > &a, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image input.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const Accumulator< A > &accu, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image input.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename I::value >::result >::ret compute_image (const Meta_Accumulator< A > &accu, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image input.
- `template<typename I , typename N , typename L >`
`mln::trait::concrete< I >::ret fill_holes (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
Filling holes of a single object in a binary image.
- `template<typename I , typename N , typename L >`
`mln::trait::ch_value< I, L >::ret flat_zones (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
Connected component labeling of the flat zones of an image.
- `template<typename I , typename N , typename L >`
`mln::trait::ch_value< I, L >::ret foreground (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I >`
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)`
Relabel a labeled image in order to have a contiguous labeling.
- `template<typename I >`
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< type-name I::value > &repack_fun)`
Relabel inplace a labeled image in order to have a contiguous labeling.
- `template<typename I , typename N , typename L >`
`mln::trait::ch_value< I, L >::ret regional_maxima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`

- `template<typename I, typename N, typename L>`
`mln::trait::ch_value< I, L >::ret regional_minima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename F>`
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)`
Remove components and relabel a labeled image.
- `template<typename I, typename F>`
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`
Remove components and relabel a labeled image.
- `template<typename I, typename F>`
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`
Remove components and relabel a labeled image inplace.
- `template<typename I, typename F>`
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2b< F > &fv2b)`
Remove components and relabel a labeled image inplace.
- `template<typename I, typename J>`
`mln::trait::concrete< I >::ret superpose (const Image< I > &lhs, const typename I::value &lhs_nlabels, const Image< J > &rhs, const typename J::value &rhs_nlabels, typename I::value &new_nlabels)`
Superpose two labeled image.
- `template<typename I, typename N, typename L>`
`mln::trait::ch_value< I, L >::ret value (const Image< I > &input, const typename I::value &val, const Neighborhood< N > &nbh, L &nlabels)`
Connected component labeling of the image sites at a given value.
- `template<typename I, typename N, typename L, typename A>`
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > value_and_compute (const Image< I > &input, const typename I::value &val, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)`
Connected component labeling of the image sites at a given value.
- `template<typename V, typename I>`
`mln::trait::ch_value< I, V >::ret wrap (const V &value_type, const Image< I > &input)`
Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).
- `template<typename I>`
`mln::trait::ch_value< I, mln::value::label_8 >::ret wrap (const Image< I > &input)`
Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

9.96.1 Detailed Description

Namespace of labeling routines.

9.96.2 Function Documentation

9.96.2.1 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::background (const Image< I > & input, const Neighborhood< N > &
nbh, L & nlabels) [inline]`

Connected component labeling of the background part in a binary image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the background.
- [out] *nlabels* The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `false`.

See also

[mln::labeling::value](#)

Definition at line 69 of file `background.hh`.

References `value()`.

Referenced by `fill_holes()`.

9.96.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::blobs (const Image< I > & input, const Neighborhood< N > & nbh, L
& nlabels) [inline]`

Connected component labeling of the binary objects of a binary image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the objects.
- [out] *nlabels* The Number of labels. Its value is set in the algorithms.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 102 of file `labeling/blobs.hh`.

Referenced by `blobs_and_compute()`, and `mln::graph::labeling()`.

9.96.2.3 `template<typename I , typename N , typename L , typename A > util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::blobs_and_compute (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Label an image and compute given accumulators.

Parameters

- [in] *input* A binary image.
- [in] *nbh* A neighborhood used for labeling.
- [in, out] *nlabels* The number of labels found.
- [in] *accu* An accumulator to be computed while labeling.

Returns

The labeled image, computed attributes for each regions and an array of the accumulators used to compute the attributes.

Definition at line 160 of file blobs_and_compute.hh.

References blobs(), and mln::make::couple().

9.96.2.4 `template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & value, const Image< L > & labeled_image, const typename L::value & nlabels) [inline]`

Create a new color image from a labeled image and fill each component with a random color.

litera::black is used for component 0, e.g. the background. Min and max values for RGB values can be set through the global variables mln::labeling::colorize_::min_value and mln::labeling::colorize_::max_value.

Parameters

- [in] *value* value type used in the returned image.
- [in] *labeled_image* A labeled image (

See also

[labeling::blobs](#)).

Parameters

- [in] *nlabels* Number of labels.

Definition at line 190 of file colorize.hh.

References mln::literal::black, and mln::data::transform().

Referenced by colorize().

9.96.2.5 `template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & value, const Image< L > & labeled_image) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 228 of file `colorize.hh`.

References `colorize()`, and `compute()`.

9.96.2.6 `template<typename L > mln::trait::ch_value< L, mln::value::rgb8 >::ret
mln::labeling::colorize (const Image< L > & input, const typename L::value & nlabels
) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 247 of file `colorize.hh`.

References `colorize()`.

9.96.2.7 `template<typename A , typename I , typename L > util::array<
mln_meta_accu_result(A, typename I::value)> mln::labeling::compute (const
Meta_Accumulator< A > & a, const Image< I > & input, const Image< L > & label,
const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

Parameters

- [in] *a* A meta-accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 734 of file `labeling/compute.hh`.

References `compute()`.

9.96.2.8 `template<typename A , typename L > util::array< typename A::result >
mln::labeling::compute (const Accumulator< A > & a, const Image< L > & label,
const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters

- [in] *a* An accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 771 of file `labeling/compute.hh`.

9.96.2.9 `template<typename A , typename L > util::array< mln_meta_accu_result(A, typename L::psite)> mln::labeling::compute (const Meta_Accumulator< A > & a, const Image< L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters

- [in] *a* A meta-accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 792 of file `labeling/compute.hh`.

References `compute()`.

9.96.2.10 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (util::array< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

Parameters

- [in] *a* An array of accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 696 of file `labeling/compute.hh`.

9.96.2.11 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (const Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

Parameters

- [in] *a* An accumulator.
- [in] *input* The input image.

[in] *label* The labeled image.
 [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 715 of file labeling/compute.hh.

Referenced by [colorize\(\)](#), [compute\(\)](#), [compute_image\(\)](#), [fill_holes\(\)](#), [mln::make::p_edges_with_mass_centers\(\)](#), [mln::make::p_vertices_with_mass_centers\(\)](#), [pack\(\)](#), and [pack_inplace\(\)](#).

9.96.2.12 `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A ::result >::ret mln::labeling::compute_image (const util::array< typename A ::result > & a, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels)`

Compute an accumulator onto the pixel values of the image *input*.
 for each component of the image *label*.

Parameters

[in] *a* The [mln::p_array](#) of accumulator result.
 [in] *input* The input image (values).
 [in] *labels* The label image.
 [in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Referenced by [compute_image\(\)](#).

9.96.2.13 `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename A ::result >::ret mln::labeling::compute_image (const Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image *input*.
 for each component of the image *label*.

Parameters

[in] *accu* The accumulator.
 [in] *input* The input image (values).
 [in] *labels* The label image.
 [in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 161 of file compute_image.hh.

References [compute\(\)](#), and [compute_image\(\)](#).

9.96.2.14 `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename
mln::internal::meta_accu_ret_result_helper< A, typename I::value >::result >::ret
mln::labeling::compute_image (const Meta_Accumulator< A > & accu, const
Image< I > & input, const Image< L > & labels, const typename L::value & nlabels
) [inline]`

Compute an accumulator onto the pixel values of the image *input*.

for each component of the image *label*.

Parameters

[in] *accu* The meta-accumulator.

[in] *input* The input image (values).

[in] *labels* The label image.

[in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 181 of file `compute_image.hh`.

References `compute()`, and `compute_image()`.

9.96.2.15 `template<typename I , typename N , typename L > mln::trait::concrete< I >::ret
mln::labeling::fill_holes (const Image< I > & input, const Neighborhood< N > &
nbh, L & nlabels) [inline]`

Filling holes of a single object in a binary image.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the background.

[out] *nlabels* The number of labels.

Returns

The binary image with a simple object without holes.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::background](#)

See also

[mln::labeling::background](#)

Definition at line 73 of file `fill_holes.hh`.

References `background()`, `compute()`, `mln::util::array< T >::nelements()`, and `mln::data::transform()`.

9.96.2.16 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::flat_zones (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the flat zones of an image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the flat zones.
- [out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 124 of file flat_zones.hh.

9.96.2.17 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::foreground (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the object part in a binary image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the foreground.
- [out] *nlabels* The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `true`.

See also

[mln::labeling::value](#)

Definition at line 69 of file foreground.hh.

References `value()`.

9.96.2.18 `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun)`

Relabel a labeled image in order to have a contiguous labeling.

Parameters

- [in] *label* The labeled image.
- [out] *new_nlabels* The number of labels after relabeling.
- [out] *repack_fun* The function used to repack the labels.

Returns

The relabeled image.

Definition at line 124 of file pack.hh.

References compute(), mln::make::relabelfun(), and mln::data::transform().

Referenced by pack().

9.96.2.19 `template<typename I> mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 115 of file pack.hh.

References pack().

9.96.2.20 `template<typename I> void mln::labeling::pack_inplace (Image< I > & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 148 of file pack.hh.

References pack_inplace().

9.96.2.21 `template<typename I> void mln::labeling::pack_inplace (Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun)`

Relabel inplace a labeled image in order to have a contiguous labeling.

Parameters

- [in] *label* The labeled image.
- [out] *new_nlabels* The number of labels after relabeling.
- [out] *repack_fun* The function used to repack the labels.

Definition at line 157 of file pack.hh.

References compute(), mln::make::relabelfun(), and mln::data::transform().

Referenced by pack_inplace().

9.96.2.22 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::regional_maxima (const Image< I > & input, const Neighborhood< N
> & nbh, L & nlabels)`

Connected component labeling of the regional maxima of an image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional maxima.
- [out] *nlabels* The number of labeled regions.

Returns

The label image.

Definition at line 130 of file regional_maxima.hh.

9.96.2.23 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::regional_minima (const Image< I > & input, const Neighborhood< N
> & nbh, L & nlabels)`

Connected component labeling of the regional minima of an image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional minima.
- [out] *nlabels* The number of labeled regions.

Returns

The label image.

Definition at line 140 of file regional_minima.hh.

Referenced by mln::morpho::meyer_wst().

9.96.2.24 `template<typename I , typename F > mln::trait::concrete< I >::ret
mln::labeling::relabel (const Image< I > & label, const typename I::value & nlabels,
typename I::value & new_nlabels, const Function_v2b< F > & fv2b) [inline]`

Remove components and relabel a labeled image.

Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [out] *new_nlabels* the number of labels after relabeling.
- [in] *fv2b* function returning whether a label must be replaced by the background.

Returns

the relabeled image.

Definition at line 200 of file relabel.hh.

References mln::make::relabelfun().

Referenced by superpose().

9.96.2.25 `template<typename I , typename F > mln::trait::concrete< I >::ret
mln::labeling::relabel (const Image< I > & label, const typename I::value & nlabels,
const Function_v2v< F > & fv2v) [inline]`

Remove components and relabel a labeled image.

Parameters

[in] *label* the labeled image.

[in] *nlabels* the number of labels in *label*.

[in] *fv2v* function returning the new component id for each pixel value.

Returns

the relabeled image.

Definition at line 179 of file relabel.hh.

References mln::data::transform().

9.96.2.26 `template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I
& label, const typename I::value & nlabels, const Function_v2v< F > & fv2v)
[inline]`

Remove components and relabel a labeled image inplace.

Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in *label*.

[in] *fv2v* function returning the new component id for each pixel value.

Definition at line 221 of file relabel.hh.

References mln::data::transform_inplace().

9.96.2.27 `template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I
& label, const typename I::value & nlabels, const Function_v2b< F > & fv2b)
[inline]`

Remove components and relabel a labeled image inplace.

Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in *label*.

[in] *fv2b* function returning whether a label must be replaced by the background.

Definition at line 240 of file relabel.hh.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

9.96.2.28 `template<typename I , typename J > mln::trait::concrete< I >::ret
mln::labeling::superpose (const Image< I > & lhs, const typename I::value &
lhs_nlabels, const Image< J > & rhs, const typename J::value & rhs_nlabels,
typename I::value & new_nlabels)`

Superpose two labeled image.

Labels in `lhs` are preserved in the output. Labels of `rhs` are renumbered from the last label value of `lhs`. It avoids duplicate label values in several components.

Parameters

- [in] *lhs* A labeled image.
- [in] *lhs_nlabels* The number of labels in `lhs`.
- [in] *rhs* A labeled image.
- [in] *rhs_nlabels* The number of labels in `rhs`.
- [out] *new_nlabels* The number of labels in the output image.

Returns

An image with all the components of `rhs` and `lhs`.

Precondition

- `rhs` and `lhs` must have the same domain.
- The value type of `rhs` must be convertible towards `lhs`'s.

Definition at line 83 of file labeling/superpose.hh.

References `mln::duplicate()`, `mln::value::equiv()`, `mln::data::paste()`, `relabel()`, and `mln::literal::zero`.

9.96.2.29 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::value (const Image< I > & input, const typename I::value & val, const
Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the image sites at a given value.

Parameters

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 149 of file labeling/value.hh.

Referenced by `background()`, and `foreground()`.

9.96.2.30 `template<typename I , typename N , typename L , typename A > util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::value_and_compute (const Image< I > & input, const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Connected component labeling of the image sites at a given value.

Parameters

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 212 of file value_and_compute.hh.

References mln::make::couple().

9.96.2.31 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::labeling::wrap (const V & value_type, const Image< I > & input) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Parameters

- [in] *value_type* The type used to wrap the label type.
- [in] *input* The label image.

Returns

A new image with values wrapped with type V.

Definition at line 75 of file labeling/wrap.hh.

References mln::data::transform().

Referenced by wrap().

9.96.2.32 `template<typename I > mln::trait::ch_value< I, mln::value::label_8 >::ret mln::labeling::wrap (const Image< I > & input) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Use label_8 as label type.

Parameters

- [in] *input* The label image.

Returns

A new image with values wrapped with type `label_8`.

Definition at line 93 of file `labeling/wrap.hh`.

References `wrap()`.

9.97 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of labeling namespace.

Functions

- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute_fastest (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`
Fastest implementation of [labeling::compute](#).
- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute_fastest (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`
Fastest implementation of [labeling::compute](#).

9.97.1 Detailed Description

Implementation namespace of labeling namespace.

9.97.2 Function Documentation

9.97.2.1 `template<typename A , typename I , typename L > util::array<typename A ::result>`
`mln::labeling::impl::compute_fastest (const Accumulator< A > & a_, const Image< I >`
`& input_, const Image< L > & label_, const typename L::value & nlabels)`
`[inline]`

Fastest implementation of [labeling::compute](#).

Parameters

- [in] *a_* An accumulator.
- [in] *input_* The input image.
- [in] *label_* The labeled image.

[in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 373 of file `labeling/compute.hh`.

References `mln::geom::ncols()`.

9.97.2.2 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::compute_fastest (util::array< A > & accus, const Image< I > &
input_, const Image< L > & label_, const typename L::value & nlabels) [inline]`

Fastest implementation of `labeling::compute`.

Parameters

[in] *accus* An array of accumulators.

[in] *input_* The input image.

[in] *label_* The labeled image.

[in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 427 of file `labeling/compute.hh`.

References `mln::geom::ncols()`, `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

Functions

- `template<typename A , typename L >
util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< L >
&label_, const typename L::value &nlabels)`
Generic implementation of `labeling::compute`.
- `template<typename A , typename L >
util::array< typename A::result > compute (util::array< A > &accus, const Image< L > &label_,
const typename L::value &nlabels)`
Generic implementation of `labeling::compute`.
- `template<typename A , typename I , typename L >
util::array< typename A::result > compute (util::array< A > &accus, const Image< I > &input_,
const Image< L > &label_, const typename L::value &nlabels)`
Generic implementation of `labeling::compute`.

- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< I >`
`&input_, const Image< L > &label_, const typename L::value &nlabels)`

Generic implementation of [labeling::compute](#).

9.98.1 Detailed Description

Generic implementation namespace of labeling namespace.

9.98.2 Function Documentation

- 9.98.2.1** `template<typename A , typename L > util::array<typename A ::result>`
`mln::labeling::impl::generic::compute (const Accumulator< A > & a_, const Image<`
`L > & label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *a_* An accumulator.
 [in] *label_* The labeled image.
 [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 204 of file `labeling/compute.hh`.

- 9.98.2.2** `template<typename A , typename L > util::array<typename A ::result>`
`mln::labeling::impl::generic::compute (util::array< A > & accus, const Image< L >`
`& label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *accus_* An array of accumulators. If the size is set to `nlabels + 1`, the accumulators are considered as initialized. Otherwise, the size is adjusted.
 [in] *label_* The labeled image.
 [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 241 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98.2.3 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (util::array< A > & accus, const Image< I > &
input_, const Image< L > & label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *accus* An array of accumulators.
- [in] *input_* The input image.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 321 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98.2.4 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (const Accumulator< A > & a_, const Image< I
> & input_, const Image< L > & label_, const typename L::value & nlabels)
[inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *a_* An accumulator.
- [in] *input_* The input image.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 283 of file `labeling/compute.hh`.

9.99 mln::linear Namespace Reference

Namespace of linear image processing routines.

Namespaces

- namespace [impl](#)
Namespace of linear image processing routines implementation details.
- namespace [local](#)
Specializations of local linear routines.

Functions

- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma)`
Gaussian filter of an image `input`.
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I , typename W , unsigned Sh, unsigned Sv>`
`mln_ch_convolve (I, W) convolve_2x1d(const Image< I > &input`
- `template<typename I , typename W >`
`mln_ch_convolve (I, W) convolve(const Image< I > &input`
- `template<typename I , typename W , unsigned S>`
`mln_ch_convolve (I, W) convolve_directional(const Image< I > &input`
- `template<typename I >`
`mln_ch_convolve_grad (I, int) sobel_2d(const Image< I > &input)`
Compute the vertical component of the 2D Sobel gradient.
- `template<typename I >`
`mln_ch_convolve (I, int) sobel_2d_h(const Image< I > &input)`
Sobel_2d gradient components.

9.99.1 Detailed Description

Namespace of linear image processing routines.

9.99.2 Function Documentation

9.99.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian (const Image< I > & input, float sigma) [inline]`

Gaussian filter of an image `input`.

Precondition

`output.domain = input.domain`

Apply an approximated gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

Precondition

`input.is_valid`

Definition at line 750 of file `gaussian.hh`.

References `mln::initialize()`.

Referenced by `mln::subsampling::gaussian_subsampling()`.

9.99.2.2 `template<typename I> mln::trait::concrete< I>::ret mln::linear::gaussian (const Image< I> & input, float sigma, int dir) [inline]`

Apply an approximated gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

`input.is_valid`
`dir < dimension(input)`

Definition at line 653 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.3 `template<typename I> mln::trait::concrete< I>::ret mln::linear::gaussian_1st_derivative (const Image< I> & input, float sigma) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input` This filter is applied in all the input image direction.

Precondition

`input.is_valid`

Definition at line 779 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.4 `template<typename I> mln::trait::concrete< I>::ret mln::linear::gaussian_1st_derivative (const Image< I> & input, float sigma, int dir) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

`input.is_valid`
`dir < dimension(input)`

Definition at line 685 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.5 `template<typename I> mln::trait::concrete< I>::ret mln::linear::gaussian_-
2nd_derivative (const Image< I> & input, float sigma)
[inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

Precondition

`input.is_valid`

Definition at line 807 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.6 `template<typename I> mln::trait::concrete< I>::ret mln::linear::gaussian_-
2nd_derivative (const Image< I> & input, float sigma, int dir)
[inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. on a specific direction `dir`. if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

`input.is_valid`
`dir < dimension(input)`

Definition at line 718 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.7 `template<typename I> mln::linear::mln_ch_convolve (I, int) const [inline]`

Sobel_2d gradient components.

Compute the L1 norm of the 2D Sobel gradient.

Compute the vertical component of the 2D Sobel gradient.

Compute the horizontal component of the 2D Sobel gradient.

Definition at line 142 of file `sobel_2d.hh`.

References `mln_ch_convolve()`, `mln_ch_convolve_grad()`, and `mln::data::transform()`.

9.99.2.8 `template<typename I , typename W , unsigned Sh, unsigned Sv>
mln::linear::mln_ch_convolve (I, W) const`

Convolution of an image `input` by two weighted line-shapes windows.

Warning

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

9.99.2.9 template<typename I , typename W > mln::linear::mln_ch_convolve (I, W) const

Convolution of an image `input` by the weighted window `w_win`.

Warning

Computation of `output (p)` is performed with the value type of `output`.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

Referenced by `mln_ch_convolve()`, and `mln_ch_convolve_grad()`.

9.99.2.10 template<typename I , typename W , unsigned S> mln::linear::mln_ch_convolve (I, W) const [inline]

Convolution of an image `input` by a line-shaped (directional) weighted window defined by the array of `weights`.

Warning

Computation of `output (p)` is performed with the value type of `output`.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

9.99.2.11 template<typename I > mln::linear::mln_ch_convolve_grad (I, int) const

Compute the vertical component of the 2D Sobel gradient.

Definition at line 124 of file `sobel_2d.hh`.

References `mln_ch_convolve()`, and `mln::data::transform()`.

Referenced by `mln_ch_convolve()`.

9.100 mln::linear::impl Namespace Reference

Namespace of linear image processing routines implementation details.

9.100.1 Detailed Description

Namespace of linear image processing routines implementation details.

9.101 mln::linear::local Namespace Reference

Specializations of local linear routines.

Namespaces

- namespace [impl](#)

Namespace of local linear routines implementation details.

Functions

- `template<typename I , typename P , typename W , typename R >`
`void convolve (const Image< I > &input, const Site< P > &p, const Weighted_Window< W > &w_win, R &result)`
- `template<typename P , typename W , typename R >`
`void convolve (const Generalized_Pixel< P > &p, const Weighted_Window< W > &w_win, R &result)`

9.101.1 Detailed Description

Specializations of local linear routines.

9.101.2 Function Documentation

9.101.2.1 `template<typename I , typename P , typename W , typename R > void`
`mln::linear::local::convolve (const Image< I > & input, const Site< P > & p, const`
`Weighted_Window< W > & w_win, R & result) [inline]`

Local convolution of image `input` at point `p` by the weighted window `w_win`.

Warning

Computation of the `result` is performed with the type `R`.
 The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 149 of file `linear/local/convolve.hh`.

Referenced by `convolve()`.

9.101.2.2 `template<typename P , typename W , typename R > void mln::linear::local::convolve (`
`const Generalized_Pixel< P > & p, const Weighted_Window< W > & w_win, R &`
`result) [inline]`

Local convolution around (generalized) pixel `b` by the weighted window `w_win`.

Warning

Computation of the `result` is performed with the type `R`.
 The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 161 of file `linear/local/convolve.hh`.

References `convolve()`.

9.102 mln::linear::local::impl Namespace Reference

Namespace of local linear routines implementation details.

9.102.1 Detailed Description

Namespace of local linear routines implementation details.

9.103 mln::literal Namespace Reference

Namespace of literals.

Classes

- struct [black_t](#)
Type of literal black.
- struct [blue_t](#)
Type of literal blue.
- struct [brown_t](#)
Type of literal brown.
- struct [cyan_t](#)
Type of literal cyan.
- struct [green_t](#)
Type of literal green.
- struct [identity_t](#)
Type of literal identity.
- struct [light_gray_t](#)
Type of literal grays.
- struct [lime_t](#)
Type of literal lime.
- struct [magenta_t](#)
Type of literal magenta.
- struct [max_t](#)
Type of literal max.
- struct [min_t](#)
Type of literal min.
- struct [olive_t](#)

Type of literal olive.

- struct [one_t](#)
Type of literal one.
- struct [orange_t](#)
Type of literal orange.
- struct [origin_t](#)
Type of literal origin.
- struct [pink_t](#)
Type of literal pink.
- struct [purple_t](#)
Type of literal purple.
- struct [red_t](#)
Type of literal red.
- struct [teal_t](#)
Type of literal teal.
- struct [violet_t](#)
Type of literal violet.
- struct [white_t](#)
Type of literal white.
- struct [yellow_t](#)
Type of literal yellow.
- struct [zero_t](#)
Type of literal zero.

Variables

- const [black_t](#) & [black](#) = [black_t](#)()
Literal black.
- const [blue_t](#) & [blue](#) = [blue_t](#)()
Literal blue.
- const [brown_t](#) & [brown](#) = [brown_t](#)()
Literal brown.
- const [cyan_t](#) & [cyan](#) = [cyan_t](#)()
Literal cyan.

- const `dark_gray_t` & `dark_gray` = `dark_gray_t()`
Literal dark gray.
- const `green_t` & `green` = `green_t()`
Literal green.
- const `identity_t` & `identity` = `identity_t()`
Literal identity.
- const `light_gray_t` & `light_gray` = `light_gray_t()`
Literal light gray.
- const `lime_t` & `lime` = `lime_t()`
Literal lime.
- const `magenta_t` & `magenta` = `magenta_t()`
Literal magenta.
- const `max_t` & `max` = `max_t()`
Literal max.
- const `medium_gray_t` & `medium_gray` = `medium_gray_t()`
Literal medium gray.
- const `min_t` & `min` = `min_t()`
Literal min.
- const `olive_t` & `olive` = `olive_t()`
Literal olive.
- const `one_t` & `one` = `one_t()`
Literal one.
- const `orange_t` & `orange` = `orange_t()`
Literal orange.
- const `origin_t` & `origin` = `origin_t()`
Literal origin.
- const `pink_t` & `pink` = `pink_t()`
Literal pink.
- const `purple_t` & `purple` = `purple_t()`
Literal purple.
- const `red_t` & `red` = `red_t()`
Literal red.
- const `teal_t` & `teal` = `teal_t()`
Literal teal.

- `const violet_t & violet = violet_t()`
Literal violet.
- `const white_t & white = white_t()`
Literal white.
- `const yellow_t & yellow = yellow_t()`
Literal yellow.
- `const zero_t & zero = zero_t()`
Literal zero.

9.103.1 Detailed Description

Namespace of literals.

9.103.2 Variable Documentation

9.103.2.1 `const black_t & mln::literal::black = black_t()`

Literal black.

Definition at line 60 of file black.hh.

Referenced by `mln::labeling::colorize()`, and `mln::registration::icp()`.

9.103.2.2 `const blue_t & mln::literal::blue = blue_t()`

Literal blue.

Definition at line 162 of file colors.hh.

9.103.2.3 `const brown_t & mln::literal::brown = brown_t()`

Literal brown.

Definition at line 164 of file colors.hh.

9.103.2.4 `const cyan_t & mln::literal::cyan = cyan_t()`

Literal cyan.

Definition at line 178 of file colors.hh.

9.103.2.5 `const dark_gray_t & mln::literal::dark_gray = dark_gray_t()`

Literal dark gray.

Definition at line 70 of file grays.hh.

9.103.2.6 const green_t & mln::literal::green = green_t()

[Literal](#) green.

Definition at line 160 of file colors.hh.

Referenced by mln::registration::icp(), and mln::make_debug_graph_image().

9.103.2.7 const identity_t & mln::literal::identity = identity_t()

[Literal](#) identity.

Definition at line 54 of file identity.hh.

9.103.2.8 const light_gray_t & mln::literal::light_gray = light_gray_t()

[Literal](#) light gray.

Definition at line 66 of file grays.hh.

9.103.2.9 const lime_t & mln::literal::lime = lime_t()

[Literal](#) lime.

Definition at line 166 of file colors.hh.

9.103.2.10 const magenta_t & mln::literal::magenta = magenta_t()

[Literal](#) magenta.

Definition at line 180 of file colors.hh.

9.103.2.11 const max_t & mln::literal::max = max_t()

[Literal](#) max.

Definition at line 66 of file literal/max.hh.

9.103.2.12 const medium_gray_t & mln::literal::medium_gray = medium_gray_t()

[Literal](#) medium_gray.

Definition at line 68 of file grays.hh.

9.103.2.13 const min_t & mln::literal::min = min_t()

[Literal](#) min.

Definition at line 66 of file literal/min.hh.

9.103.2.14 `const olive_t & mln::literal::olive = olive_t()`

[Literal](#) olive.

Definition at line 184 of file colors.hh.

9.103.2.15 `const one_t & mln::literal::one = one_t()`

[Literal](#) one.

Definition at line 69 of file one.hh.

Referenced by `mln::algebra::h_vec< d, C >::h_vec()`, `mln::operator++()`, and `mln::operator--()`.

9.103.2.16 `const orange_t & mln::literal::orange = orange_t()`

[Literal](#) orange.

Definition at line 168 of file colors.hh.

9.103.2.17 `const origin_t & mln::literal::origin = origin_t()`

[Literal](#) origin.

Definition at line 55 of file origin.hh.

Referenced by `mln::win::ball< G, C >::ball()`, `mln::geom::bbox()`, `mln::box< P >::box()`, `mln::geom::rotate()`, and `mln::make::w_window()`.

9.103.2.18 `const pink_t & mln::literal::pink = pink_t()`

[Literal](#) pink.

Definition at line 170 of file colors.hh.

9.103.2.19 `const purple_t & mln::literal::purple = purple_t()`

[Literal](#) purple.

Definition at line 172 of file colors.hh.

9.103.2.20 `const red_t & mln::literal::red = red_t()`

[Literal](#) red.

Definition at line 158 of file colors.hh.

Referenced by `mln::morpho::watershed::superpose()`, and `mln::debug::superpose()`.

9.103.2.21 `const teal_t & mln::literal::teal = teal_t()`

[Literal](#) teal.

Definition at line 174 of file colors.hh.

9.103.2.22 const violet_t & mln::literal::violet = violet_t()

[Literal](#) violet.

Definition at line 176 of file colors.hh.

9.103.2.23 const white_t & mln::literal::white = white_t()

[Literal](#) white.

Definition at line 60 of file white.hh.

Referenced by mln::registration::icp().

9.103.2.24 const yellow_t & mln::literal::yellow = yellow_t()

[Literal](#) yellow.

Definition at line 182 of file colors.hh.

9.103.2.25 const zero_t & mln::literal::zero = zero_t()

[Literal](#) zero.

Definition at line 69 of file zero.hh.

Referenced by mln::transform::influence_zone_geodesic_saturated(), mln::accu::shape::volume< I >::init(), mln::accu::stat::variance< T, S, R >::init(), mln::morpho::attribute::sum< I, S >::init(), mln::accu::math::sum< T, S >::init(), mln::accu::rms< T, V >::init(), mln::accu::stat::histo3d_rgb< V >::init(), mln::accu::convolve< T1, T2, R >::init(), mln::accu::center< P, V >::init(), mln::window< D >::is_centered(), mln::accu::stat::variance< T, S, R >::mean(), mln::accu::stat::var< T >::mean(), mln::geom::mesh_corner_point_area(), mln::geom::mesh_curvature(), mln::geom::mesh_normal(), mln::morpho::meyer_wst(), mln::algebra::operator*(), mln::test::positive(), mln::make::relabelfun(), mln::geom::rotate(), mln::accu::shape::volume< I >::set_value(), mln::morpho::watershed::superpose(), mln::labeling::superpose(), mln::debug::superpose(), mln::accu::stat::var< T >::to_result(), mln::geom::translate(), and mln::make::w_window_directional().

9.104 mln::logical Namespace Reference

Namespace of logic.

Namespaces

- namespace [impl](#)

Implementation namespace of logical namespace.

Functions

- template<typename L, typename R>
void [and_inplace](#) ([Image](#)< L > &lhs, const [Image](#)< R > &rhs)

- `template<typename L , typename R >`
`mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret and_not (const Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L , typename R >`
`void and_not_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename I >`
`void not_inplace (Image< I > &input)`
- `template<typename L , typename R >`
`void or_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L , typename R >`
`void xor_inplace (Image< L > &lhs, const Image< R > &rhs)`

9.104.1 Detailed Description

Namespace of logic.

9.104.2 Function Documentation

9.104.2.1 `template<typename L , typename R > void mln::logical::and_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical and" of image `rhs` in image `lhs`.

Parameters

- `[in, out]` *lhs* First operand image.
`[in]` *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/and.hh`.

References `mln::data::transform_inplace()`.

9.104.2.2 `template<typename L , typename R > mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret mln::logical::and_not (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise "logical and-not" between images `lhs` and `rhs`.

Parameters

- `[in]` *lhs* First operand image.
`[in]` *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 76 of file and_not.hh.

References mln::data::transform().

9.104.2.3 `template<typename L , typename R > void mln::logical::and_not_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical and-not" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and not rhs(p)`

Precondition

```
rhs.domain >= lhs.domain
```

Definition at line 91 of file and_not.hh.

References mln::data::transform_inplace().

9.104.2.4 `template<typename I > void mln::logical::not_inplace (Image< I > & input) [inline]`

Point-wise in-place "logical not" of image `input`.

Parameters

[in, out] *input* The target image.

It performs:

for all `p` of `input.domain`

`input(p) = not input(p)`

Precondition

```
input.is_valid
```

Definition at line 88 of file logical/not.hh.

References mln::data::transform_inplace().

9.104.2.5 `template<typename L , typename R > void mln::logical::or_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical or" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.
 [in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) or rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/or.hh`.

References `mln::data::transform_inplace()`.

9.104.2.6 `template<typename L , typename R > void mln::logical::xor_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical xor" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.
 [in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) xor rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/xor.hh`.

References `mln::data::transform_inplace()`.

9.105 `mln::logical::impl` Namespace Reference

Implementation namespace of logical namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of logical namespace.

9.105.1 Detailed Description

Implementation namespace of logical namespace.

9.106 mln::logical::impl::generic Namespace Reference

Generic implementation namespace of logical namespace.

9.106.1 Detailed Description

Generic implementation namespace of logical namespace.

9.107 mln::make Namespace Reference

Namespace of routines that help to make Milena's objects.

Functions

- `template<unsigned D, typename G, typename V>`
`p_set< complex_psite< D, G > > attachment (const complex_psite< D, G > &f, const complex_-`
`image< D, G, V > &ima)`
Compute the attachment of the cell corresponding to the facet f to the image ima.
- `mln::box1d box1d (def::coord min_ind, def::coord max_ind)`
Create an mln::box1d.
- `mln::box1d box1d (unsigned ninds)`
Create an mln::box1d.
- `mln::box2d box2d (unsigned nrows, unsigned ncols)`
Create an mln::box2d.
- `mln::box2d box2d (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_-`
`col)`
Create an mln::box2d.
- `mln::box2d_h box2d_h (def::coord min_row, def::coord min_col, def::coord max_row, def::coord`
`max_col)`
Create an mln::box2d_h.
- `mln::box2d_h box2d_h (unsigned nrows, unsigned ncols)`
Create an mln::box2d_h.
- `mln::box3d box3d (unsigned nslis, unsigned nrows, unsigned ncols)`
Create an mln::box3d.

- `mln::box3d box3d (def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col)`
Create an `mln::box3d`.
- `template<unsigned D, typename G >`
`p_set< complex_psite< D, G > > cell (const complex_psite< D, G > &f)`
Compute the set of faces of the cell corresponding to the facet `f`.
- `template<typename T, typename U >`
`util::couple< T, U > couple (const T &val1, const T &val2)`
Construct an `mln::util::couple` on-the-fly.
- `template<unsigned D, typename G, typename V >`
`p_set< complex_psite< D, G > > detachment (const complex_psite< D, G > &f, const complex_image< D, G, V > &ima)`
Compute the detachment of the cell corresponding to the facet `f` to the image `ima`.
- `mln::dpoint2d_h dpoint2d_h (def::coord row, def::coord col)`
Create an `mln::dpoint2d_h`.
- `template<typename G, typename P >`
`p_edges< G, pw::cst_< P > > dummy_p_edges (const Graph< G > &g, const P &dummy_site)`
Create a `p_edges` which associate a graph element to a constant site.
- `template<typename G >`
`p_edges< G > dummy_p_edges (const Graph< G > &g)`
Create a `p_edges` which associate a graph element to a constant site.
- `template<typename G, typename P >`
`p_vertices< G, pw::cst_< P > > dummy_p_vertices (const Graph< G > &g, const P &dummy_site)`
Create a `p_vertices` which associate a graph element to a constant site.
- `template<typename G >`
`p_vertices< G > dummy_p_vertices (const Graph< G > &g)`
Create a `p_vertices` which associate a graph element to a constant site.
- `template<typename V, typename G >`
`mln::edge_image< void, V, G > edge_image (const Graph< G > &g, const fun::i2v::array< V > &fv)`
Construct an edge image.
- `template<typename FV, typename G >`
`mln::edge_image< void, typename FV::result, G > edge_image (const Graph< G > &g, const Function_v2v< FV > &fv)`
Construct an edge image.
- `template<typename FP, typename FV, typename G >`
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const Graph< G > &g, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`
Construct an edge image.

- `template<typename P, typename V, typename G, typename FP, typename FV >`
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > &fv_)`
Construct an edge image.
- `template<typename P, typename V, typename G, typename FV >`
`mln::edge_image< void, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const Function_vv2v< FV > &fv_)`
Construct an edge image.
- `template<typename P, typename V, typename G, typename F >`
`mln::edge_image< void, bool, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const Function_v2b< F > &fv_)`
Construct an edge image.
- `template<typename T, unsigned N>`
`algebra::h_mat< mlc_sqrt_int(N), T > h_mat (const T(&tab)[N])`
Create an mln::algebra::mat<n,n,T>.
- `template<typename V, unsigned L>`
`mln::image1d< V > image (V(&values)[L])`
Create an image1d from an 1D array of values.
- `template<typename V, unsigned R, unsigned C>`
`mln::image2d< V > image (V(&values)[R][C])`
Create an image2d from an 2D array of values.
- `template<typename V, unsigned S, unsigned R, unsigned C>`
`mln::image3d< V > image (V(&values)[S][R][C])`
Create an image3d from an 3D array of values.
- `template<typename V, unsigned S>`
`mln::image2d< V > image2d (V(&values)[S])`
Create an image2d from an 2D array of values.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const Image< I > &ima)`
Create an image3d from a 2D image.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const util::array< I > &ima)`
Create an image3d from an array of 2D images.
- `template<typename I, typename N >`
`util::graph_influence_zone_adjacency_graph (const Image< I > &iz_, const Neighborhood< N > &nbh, const typename I::value &nlabels)`
Create a graph from an influence zone image.
- `template<unsigned n, unsigned m, typename T >`
`algebra::mat< n, m, T > mat (const T(&tab)[n * m])`

Create an `mln::algebra::mat<n,m,T>`.

- `template<typename T >`
`util::ord_pair< T > ord_pair (const T &val1, const T &val2)`
 Construct an `mln::util::ord_pair` on-the-fly.
- `template<typename W , typename G >`
`p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > p_edges_with_mass_centers`
`(const Image< W > &wst_, const Graph< G > &g_)`
 Construct a `p_edges` from a watershed image and a region adjacency graph (RAG).
- `template<typename W , typename G >`
`p_vertices< G, fun::i2v::array< typename W::site > > p_vertices_with_mass_centers (const Image< W > &wst_, const Graph< G > &g_)`
 Construct a `p_vertices` from a watershed image and a region adjacency graph (RAG).
- `template<typename I >`
`mln::util::pix< I > pix (const Image< I > &ima, const typename I::psite &p)`
 Create an `mln::util::pix` from an image `ima` and a psite `p`.
- `template<typename I >`
`mln::pixel< I > pixel (Image< I > &ima, const typename I::psite &p)`
 Create a `mln::pixel` from a mutable image `ima` and a point `p`.
- `template<typename I >`
`mln::pixel< const I > pixel (const Image< I > &ima, const typename I::psite &p)`
 Create a `mln::pixel` from a constant image `ima` and a point `p`.
- `mln::point2d_h point2d_h (def::coord row, def::coord col)`
 Create an `mln::point2d_h`.
- `template<typename I , typename N >`
`util::couple< util::graph, typename mln::trait::concrete< I >::ret > rag_and_labeled_wsl (const Image< I > &wshd_, const Neighborhood< N > &nbh_, const typename I::value &nbasins)`
 Create a region adjacency graph and a label image of the watershed line from a watershed image.
- `template<typename I , typename N >`
`util::graph region_adjacency_graph (const Image< I > &wshd_, const Neighborhood< N > &nbh, const typename I::value &nbasins)`
 Create a region adjacency graph from a watershed image.
- `template<typename V , typename F >`
`fun::i2v::array< V > relabelfun (const Function_v2b< F > &fv2b, const V &nlabels, V &new_nlabels)`
 Create a `i2v` function from a `v2b` function.
- `template<typename V , typename F >`
`fun::i2v::array< V > relabelfun (const Function_v2v< F > &fv2v, const V &nlabels, V &new_nlabels)`
 Create a `i2v` function from a `v2v` function.

- `template<typename T >`
`algebra::vec< 2, T > vec (const T &v_0, const T &v_1)`
Create an `mln::algebra::vec<2,T>`.
- `template<typename T >`
`algebra::vec< 4, T > vec (const T &v_0, const T &v_1, const T &v_2, const T &v_3)`
Create an `mln::algebra::vec<4,T>`.
- `template<typename T >`
`algebra::vec< 3, T > vec (const T &v_0, const T &v_1, const T &v_2)`
Create an `mln::algebra::vec<3,T>`.
- `template<typename T >`
`algebra::vec< 1, T > vec (const T &v_0)`
Create an `mln::algebra::vec<n,T>`.
- `template<typename FP , typename FV , typename G >`
`mln::vertex_image< typename FP::result, typename FV::result, G > vertex_image (const Graph< G > &g_, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`
Construct a vertex image.
- `template<typename G , typename FV >`
`mln::vertex_image< void, typename FV::result, G > vertex_image (const Graph< G > &g, const Function_v2v< FV > &fv)`
Construct a vertex image.
- `template<typename I , typename N >`
`p_vertices< util::graph, fun::i2v::array< typename I::site > > voronoi (Image< I > &ima_, Image< I > &orig_, const Neighborhood< N > &nbh)`
Apply the Voronoi algorithm on `ima_` with the original image `orig_` for node computing with neighborhood `nbh`.
- `template<typename W , typename F >`
`mln::w_window< typename W::dpsite, typename F::result > w_window (const Window< W > &win, const Function_v2v< F > &wei)`
Create a `mln::w_window` from a window and a weight function.
- `template<typename W , unsigned M>`
`mln::w_window< mln::dpoint1d, W > w_window1d (W(&weights)[M])`
Create a 1D `mln::w_window` from an array of weights.
- `template<typename W , unsigned S>`
`mln::w_window< mln::dpoint2d, W > w_window2d (W(&weights)[S])`
Create a 2D `mln::w_window` from an array of weights.
- `template<typename W , unsigned M>`
`mln::w_window< mln::dpoint3d, W > w_window3d (W(&weights)[M])`
Create a 3D `mln::w_window` from an array of weights.
- `template<typename D , typename W , unsigned L>`
`mln::w_window< D, W > w_window_directional (const Gdpoint< D > &dp, W(&weights)[L])`
Create a directional centered weighted window.

9.107.1 Detailed Description

Namespace of routines that help to make Milena's objects.

9.107.2 Function Documentation

9.107.2.1 `template<unsigned D, typename G , typename V > p_set< complex_psite< D, G > >
mln::make::attachment (const complex_psite< D, G > & f, const complex_image< D,
G, V > & ima) [inline]`

Compute the attachment of the cell corresponding to the facet f to the image ima .

Precondition

f is a facet (it does not belong to any face of higher dimension).
 ima is an image of Boolean values.

Returns

a set of faces containing the attachment.

We do not use the formal definition of the attachment here (see `couprie.08.pami`). We use the following (equivalent) definition: an N -face F in $CELL$ is in the attachment of $CELL$ to IMA if it is adjacent to at least an $(N-1)$ -face or an $(N+1)$ -face that does not belong to $CELL$.

Definition at line 68 of file `attachment.hh`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()`.

9.107.2.2 `mln::box1d mln::make::box1d (def::coord min_ind, def::coord max_ind)
[inline]`

Create an [mln::box1d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

[in] *min_ind* Minimum index.

[in] *max_ind* Maximum index.

Precondition

`max_ind >= min_ind`.

Returns

A 1D box.

Definition at line 79 of file `make/box1d.hh`.

References `box1d()`.

9.107.2.3 mln::box1d mln::make::box1d (unsigned *ninds*) [inline]

Create an [mln::box1d](#).

Parameters

[in] *ninds* Number of indices.

Precondition

ninds != 0 and *ncols* != 0.

Returns

A 1D box.

Definition at line 70 of file make/box1d.hh.

Referenced by `box1d()`, and `mln::image1d< T >::image1d()`.

9.107.2.4 mln::box2d mln::make::box2d (unsigned *nrows*, unsigned *ncols*) [inline]

Create an [mln::box2d](#).

Parameters

[in] *nrows* Number of rows.

[in] *ncols* Number of columns.

Precondition

nrows != 0 and *ncols* != 0.

Returns

A 2D box.

Definition at line 78 of file make/box2d.hh.

Referenced by `mln::image2d< T >::image2d()`, and `mln::io::pnm::load()`.

9.107.2.5 mln::box2d mln::make::box2d (def::coord *min_row*, def::coord *min_col*, def::coord *max_row*, def::coord *max_col*) [inline]

Create an [mln::box2d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

[in] *min_row* Index of the top most row.

[in] *min_col* Index of the left most column.

[in] *max_row* Index of the bottom most row.

[in] *max_col* Index of the right most column.

Precondition

`max_row >= min_row and max_col >= min_col.`

Returns

A 2D box.

Definition at line 88 of file `make/box2d.hh`.

9.107.2.6 `mln::box2d_h mln::make::box2d_h (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col) [inline]`

Create an [mln::box2d_h](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *min_row* Index of the top most row.
- [in] *min_col* Index of the left most column.
- [in] *max_row* Index of the bottom most row.
- [in] *max_col* Index of the right most column.

Precondition

`max_row >= min_row and max_col >= min_col.`

Returns

A 2D_H box.

Definition at line 85 of file `make/box2d_h.hh`.

References `point2d_h()`.

9.107.2.7 `mln::box2d_h mln::make::box2d_h (unsigned nrows, unsigned ncols) [inline]`

Create an [mln::box2d_h](#).

Parameters

- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

Precondition

`nrows != 0 and ncols != 0.`

Returns

A 2D_H box.

Definition at line 75 of file `make/box2d_h.hh`.

References `point2d_h()`.

9.107.2.8 mln::box3d mln::make::box3d (unsigned *nslis*, unsigned *nrows*, unsigned *ncols*) [inline]

Create an [mln::box3d](#).

Parameters

- [in] *nslis* Number of slices.
- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

Precondition

`ninds != 0 and ncols != 0 and nslis != 0.`

Returns

A 3D box.

Definition at line 80 of file `make/box3d.hh`.

Referenced by `image3d()`, and `mln::image3d< T >::image3d()`.

9.107.2.9 mln::box3d mln::make::box3d (def::coord *min_sli*, def::coord *min_row*, def::coord *min_col*, def::coord *max_sli*, def::coord *max_row*, def::coord *max_col*) [inline]

Create an [mln::box3d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *min_sli* Index of the lowest slice.
- [in] *min_row* Index of the top most row.
- [in] *min_col* Index of the left most column.
- [in] *max_sli* Index of the highest slice.
- [in] *max_row* Index of the botton most row.
- [in] *max_col* Index of the right most column.

Precondition

```
max_sli >= min_sli.
max_row >= min_row.
max_col >= min_col.
```

Returns

A 3D box.

Definition at line 92 of file `make/box3d.hh`.

9.107.2.10 `template<unsigned D, typename G > p_set< complex_psite< D, G > >
mln::make::cell (const complex_psite< D, G > & f) [inline]`

Compute the set of faces of the cell corresponding to the facet *f*.

Precondition

f is a facet (it does not belong to any face of higher dimension).

Returns

An [mln::p_set](#) of sites (faces) containing the attachment.

Definition at line 63 of file cell.hh.

References [mln::topo::is_facet\(\)](#), and [mln::complex_psite< D, G >::n\(\)](#).

Referenced by [attachment\(\)](#), and [detachment\(\)](#).

9.107.2.11 `template<typename T , typename U > util::couple<T,U> mln::make::couple (const
T & val1, const T & val2)`

Construct an [mln::util::couple](#) on-the-fly.

Referenced by [mln::labeling::blobs_and_compute\(\)](#), [mln::transform::distance_and_closest_point_geodesic\(\)](#), [mln::transform::distance_and_influence_zone_geodesic\(\)](#), and [mln::labeling::value_and_compute\(\)](#).

9.107.2.12 `template<unsigned D, typename G , typename V > p_set< complex_psite< D, G > >
mln::make::detachment (const complex_psite< D, G > & f, const complex_image<
D, G, V > & ima) [inline]`

Compute the detachment of the cell corresponding to the facet *f* to the image *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).
ima is an image of Boolean values.

Returns

a set of faces containing the detachment.

We do not use the formal definition of the detachment here (see [coupric.08.pami](#)). We use the following (equivalent) definition: an N-face F in CELL is not in the detachment of CELL from IMA if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to CELL.

Definition at line 68 of file detachment.hh.

References [cell\(\)](#), and [mln::topo::is_facet\(\)](#).

Referenced by [mln::topo::detach\(\)](#).

9.107.2.13 `mln::dpoint2d_h mln::make::dpoint2d_h (def::coord row, def::coord col)
[inline]`

Create an [mln::dpoint2d_h](#).

Parameters

- [in] *row* Row coordinate.
 [in] *col* Column coordinate.

Returns

A 2D dpoint.

Definition at line 55 of file make/dpoint2d_h.hh.

9.107.2.14 `template<typename G , typename P > p_edges< G, pw::cst_< P > >
 mln::make::dummy_p_edges (const Graph< G > & g_, const P & dummy_site)`

Create a [p_edges](#) which associate a graph element to a constant site.

Parameters

- [in] *g_* A graph.
 [in] *dummy_site* The dummy site mapped to graph edges.

Returns

A [p_edges](#).

Definition at line 77 of file dummy_p_edges.hh.

9.107.2.15 `template<typename G > p_edges< G > mln::make::dummy_p_edges (const Graph<
 G > & g)`

Create a [p_edges](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

- [in] *g* A graph.

Returns

A [p_edges](#).

Definition at line 93 of file dummy_p_edges.hh.

9.107.2.16 `template<typename G , typename P > p_vertices< G, pw::cst_< P > >
 mln::make::dummy_p_vertices (const Graph< G > & g_, const P & dummy_site)`

Create a [p_vertices](#) which associate a graph element to a constant site.

Parameters

- [in] *g_* A graph.
 [in] *dummy_site* The dummy site mapped to graph vertices.

Returns

A [p_vertices](#).

Definition at line 77 of file dummy_p_vertices.hh.

9.107.2.17 `template<typename G > p_vertices< G > mln::make::dummy_p_vertices (const Graph< G > & g)`

Create a [p_vertices](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

[in] *g* A graph.

Returns

A [p_vertices](#).

Definition at line 93 of file dummy_p_vertices.hh.

9.107.2.18 `template<typename V , typename G > mln::edge_image< void, V, G > mln::make::edge_image (const Graph< G > & g, const fun::i2v::array< V > & fv) [inline]`

Construct an edge image.

Parameters

[in] *g* A graph.

[in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 141 of file make/edge_image.hh.

9.107.2.19 `template<typename FV , typename G > mln::edge_image< void, typename FV::result, G > mln::make::edge_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct an edge image.

Parameters

[in] *g* A graph.

[in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 155 of file make/edge_image.hh.

9.107.2.20 `template<typename FP , typename FV , typename G > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image (const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv) [inline]`

Construct an edge image.

Parameters

- [in] *g_* A graph.
- [in] *fp* A function mapping edge ids to sites.
- [in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 179 of file make/edge_image.hh.

9.107.2.21 `template<typename P , typename V , typename G , typename FP , typename FV > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > & fv_) [inline]`

Construct an edge image.

Parameters

- [in] *v_ima_* A vertex image.
- [in] *pe* A [p_edges](#) mapping graph elements to sites.
- [in] *fv_* A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

Returns

an edge image.

Definition at line 199 of file make/edge_image.hh.

9.107.2.22 `template<typename P , typename V , typename G , typename FV > mln::edge_image< void, typename FV::result, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_, const Function_vv2v< FV > & fv_) [inline]`

Construct an edge image.

Parameters

- [in] *v_ima_* A vertex image.
- [in] *fv_* A function mapping two vertices' values to a value. The result is associated to the corresponding edge.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 225 of file make/edge_image.hh.

9.107.2.23 `template<typename P , typename V , typename G , typename F > mln::edge_image< void, bool, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_, const Function_v2b< F > & fv_) [inline]`

Construct an edge image.

Parameters

[in] *v_ima_* A vertex image.

[in] *fv_* A predicate on a vertex's value. The (Boolean) result is associated to the edges adjacent to the vertex.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 250 of file make/edge_image.hh.

References `mln::data::fill()`.

9.107.2.24 `template<typename T , unsigned N> algebra::h_mat< mlc_sqrt_int(N), T > mln::make::h_mat (const T(&) tab[N]) [inline]`

Create an `mln::algebra::mat<n,n,T>`.

Definition at line 60 of file make/h_mat.hh.

Referenced by `mln::fun::x2x::rotation< n, C >::rotation()`.

9.107.2.25 `template<typename V , unsigned L> mln::image1d< V > mln::make::image (V(&) values[L])`

Create an [image1d](#) from an 1D array of values.

Parameters

[in] *values* 1D array.

Returns

A 1D image.

Definition at line 85 of file make/image.hh.

9.107.2.26 `template<typename V , unsigned R, unsigned C> mln::image2d< V > mln::make::image (V(&) values[R][C])`

Create an [image2d](#) from an 2D array of values.

Parameters

[in] *values* 2D array.

Returns

A 2D image.

Definition at line 97 of file make/image.hh.

References mln::opt::at().

9.107.2.27 `template<typename V , unsigned S, unsigned R, unsigned C> mln::image3d< V >
mln::make::image (V(&) values[S][R][C])`

Create an [image3d](#) from an 3D array of values.

Parameters

[in] *values* 3D array.

Returns

A 3D image.

Definition at line 112 of file make/image.hh.

References mln::opt::at().

9.107.2.28 `template<typename V , unsigned S> mln::image2d< V > mln::make::image2d (V(&)
values[S])`

Create an [image2d](#) from an 2D array of values.

Parameters

[in] *values* 2D array.

Returns

A 2D image.

Definition at line 61 of file make/image2d.hh.

9.107.2.29 `template<typename I > mln::image3d< typename I::value > mln::make::image3d (
const Image< I > & ima) [inline]`

Create an [image3d](#) from a 2D image.

Definition at line 92 of file make/image3d.hh.

References box3d(), and mln::data::paste().

9.107.2.30 `template<typename I > mln::image3d< typename I::value > mln::make::image3d (
const util::array< I > & ima) [inline]`

Create an [image3d](#) from an array of 2D images.

Definition at line 70 of file make/image3d.hh.

References box3d(), mln::util::array< T >::is_empty(), mln::util::array< T >::nelements(), mln::data::paste(), mln::box< P >::pmax(), and mln::box< P >::pmin().

Referenced by mln::io::pnms::load().

9.107.2.31 `template<typename I , typename N > util::graph mln::make::influence_zone_adjacency_graph (const Image< I > & iz, const Neighborhood< N > & nbh, const typename I::value & nlabels) [inline]`

Create a graph from an influence zone image.

Parameters

- [in] *iz* influence zone image.
- [in] *nbh* A neighborhood.
- [in] *nlabels* number of influence zone in *iz*.

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 175 of file `influence_zone_adjacency_graph.hh`.

9.107.2.32 `template<unsigned n, unsigned m, typename T > algebra::mat< n, m, T > mln::make::mat (const T(&) tab[n *m]) [inline]`

Create an `mln::algebra::mat<n,m,T>`.

Parameters

- [in] *tab* Array of values.

Precondition

The array dimension has to be $n * m$.

Definition at line 61 of file `make/mat.hh`.

9.107.2.33 `template<typename T > util::ord_pair< T > mln::make::ord_pair (const T & val1, const T & val2) [inline]`

Construct an [mln::util::ord_pair](#) on-the-fly.

Definition at line 277 of file `ord_pair.hh`.

9.107.2.34 `template<typename W , typename G > p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > mln::make::p_edges_with_mass_centers (const Image< W > & wst, const Graph< G > & g) [inline]`

Construct a [p_edges](#) from a watershed image and a region adjacency graph (RAG).

Map each graph edge to a pair of mass centers of two adjacent regions.

Parameters

- wst* A watershed image.
- g* A region adjacency graph.

Returns

A [p_edges](#).

See also

[edge_image](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 81 of file `p_edges_with_mass_centers.hh`.

References `mln::labeling::compute()`.

9.107.2.35 `template<typename W , typename G > p_vertices< G, fun::i2v::array< typename W::site > > mln::make::p_vertices_with_mass_centers (const Image< W > & wst_, const Graph< G > & g_) [inline]`

Construct a [p_vertices](#) from a watershed image and a region adjacency graph (RAG).

Map each graph vertex to the mass center of its corresponding region.

Parameters

wst_ A watershed image.

g_ A region adjacency graph.

Returns

A [p_vertices](#).

See also

[edge_image](#), [vertex_image](#), [p_vertices](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 77 of file `p_vertices_with_mass_centers.hh`.

References `mln::labeling::compute()`.

9.107.2.36 `template<typename I > mln::util::pix< I > mln::make::pix (const Image< I > & ima, const typename I::psite & p) [inline]`

Create an [mln::util::pix](#) from an image *ima* and a psite *p*.

Parameters

[in] *ima* The input image.

[in] *p* The point site.

Returns

An [mln::util::pix](#).

Definition at line 58 of file `make/pix.hh`.

9.107.2.37 `template<typename I > mln::pixel< I > mln::make::pixel (Image< I > & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a mutable image *ima* and a point *p*.

Definition at line 63 of file `make/pixel.hh`.

9.107.2.38 `template<typename I> mln::pixel< const I> mln::make::pixel (const Image< I> & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a constant image `ima` and a point `p`.

Definition at line 55 of file `make/pixel.hh`.

9.107.2.39 `mln::point2d_h mln::make::point2d_h (def::coord row, def::coord col) [inline]`

Create an [mln::point2d_h](#).

Parameters

- [in] *row* Row coordinate.
- [in] *col* Column coordinate.

Returns

A 2D point.

Definition at line 55 of file `make/point2d_h.hh`.

Referenced by `box2d_h()`.

9.107.2.40 `template<typename I, typename N> util::couple< util::graph, typename mln::trait::concrete< I>::ret> mln::make::rag_and_labeled_wsl (const Image< I> & wshd_, const Neighborhood< N> & nbh_, const typename I::value & nbasins) [inline]`

Create a region adjacency graph and a label image of the watershed line from a watershed image.

Parameters

- [in] *wshd_* Watershed image.
- [in] *nbh_* [Neighborhood](#)
- [in] *nbasins* Number of influence zone in *wshd*.

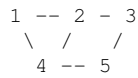
Returns

A couple. First element is the graph, second element is an image with a labeled watershed line.

<pre> ----- 1 1 1 0 2 2 0 3 1 1 0 2 2 2 0 3 1 0 4 0 2 0 3 3 0 4 4 4 0 5 0 3 ----- </pre>	<pre> -----> </pre>	<pre> ----- . . . 1 . . 2 . . . 1 . . . 2 . . 1 . 3 . 4 . . 1 . . . 5 . 6 . ----- </pre>
--	------------------------	--

Watershed image Labeled watershed line
(watershed line labeled with 0)

|
|
|
v



Region Adjacency graph (RAG)

Definition at line 229 of file rag_and_labeled_wsl.hh.

9.107.2.41 `template<typename I, typename N> util::graph mln::make::region_adjacency_graph (const Image< I> & wshd_, const Neighborhood< N> & nbh, const typename I::value & nbasins) [inline]`

Create a region adjacency graph from a watershed image.

Parameters

- [in] *wshd_* watershed image.
- [in] *nbh* A neighborhood.
- [in] *nbasins* number of influence zone in wshd.

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 179 of file region_adjacency_graph.hh.

9.107.2.42 `template<typename V, typename F> fun::i2v::array< V> mln::make::relabelfun (const Function_v2b< F> & fv2b, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2b function.

This function can be used to relabel a labeled image.

Parameters

- [in] *fv2b* A v2b function.
- [in] *nlabels* The number of labels.
- [in] *new_nlabels* The number of labels after relabeling.

Returns

a i2v function.

See also

[mln::labeling::relabel](#)

Definition at line 83 of file relabelfun.hh.

References `mln::literal::zero`.

Referenced by `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `mln::labeled_image_base< I, E>::relabel()`, and `mln::labeling::relabel_inplace()`.

9.107.2.43 `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun (const Function_v2v< F > & fv2v, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2v function.

This function can be used to relabel a labeled image.

Parameters

[in] *fv2v* A v2v function. This function maps an id to an already existing one.

[in] *nlabels* The number of labels.

[in] *new_nlabels* The number of labels after relabeling.

Returns

a i2v function.

See also

[mln::labeling::relabel](#)

Definition at line 106 of file relabelfun.hh.

References mln::literal::zero.

9.107.2.44 `template<typename T > algebra::vec< 2, T > mln::make::vec (const T & v_0, const T & v_1) [inline]`

Create an mln::algebra::vec<2,T>.

Parameters

[in] *v_0* First coordinate.

[in] *v_1* Second coordinate.

Returns

A 2D vector.

Definition at line 100 of file make/vec.hh.

9.107.2.45 `template<typename T > algebra::vec< 4, T > mln::make::vec (const T & v_0, const T & v_1, const T & v_2, const T & v_3) [inline]`

Create an mln::algebra::vec<4,T>.

Parameters

[in] *v_0* First coordinate.

[in] *v_1* Second coordinate.

[in] *v_2* Third coordinate.

[in] *v_3* Fourth coordinate.

Returns

A 4D vector.

Definition at line 121 of file make/vec.hh.

9.107.2.46 `template<typename T> algebra::vec< 3, T> mln::make::vec (const T & v_0, const T & v_1, const T & v_2) [inline]`

Create an mln::algebra::vec<3,T>.

Parameters

[in] *v_0* First coordinate.

[in] *v_1* Second coordinate.

[in] *v_2* Third coordinate.

Returns

A 3D vector.

Definition at line 110 of file make/vec.hh.

9.107.2.47 `template<typename T> algebra::vec< 1, T> mln::make::vec (const T & v_0) [inline]`

Create an mln::algebra::vec<n,T>.

Parameters

[in] *v_0* First coordinate.

Returns

A 1D vector.

Definition at line 91 of file make/vec.hh.

9.107.2.48 `template<typename FP , typename FV , typename G> mln::vertex_image< typename FP::result, typename FV::result, G> mln::make::vertex_image (const Graph< G> & g_, const Function_v2v< FP> & fp, const Function_v2v< FV> & fv)`

Construct a vertex image.

Parameters

[in] *g_* A graph.

[in] *fp* A function mapping vertex ids to sites.

[in] *fv* A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 92 of file make/vertex_image.hh.

9.107.2.49 `template<typename G , typename FV > mln::vertex_image< void, typename FV::result, G > mln::make::vertex_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct a vertex image.

Parameters

- [in] *g* A graph.
- [in] *fv* A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 77 of file make/vertex_image.hh.

9.107.2.50 `template<typename I , typename N > p_vertices< util::graph, fun::i2v::array< typename I::site > > mln::make::voronoi (Image< I > & ima_, Image< I > & orig_, const Neighborhood< N > & nbh) [inline]`

Apply the Voronoi algorithm on *ima_* with the original image *orig_* for node computing with neighborhood *nbh*.

Parameters

- [in] *ima_* The labeling image.
- [in] *orig_* The original image.
- [in] *nbh* The neighborhood for computing algorithm.

Returns

The computed graph.

Definition at line 68 of file voronoi.hh.

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

9.107.2.51 `template<typename W , typename F > mln::w_window< typename W::dpsite, typename F::result > mln::make::w_window (const Window< W > & win, const Function_v2v< F > & wei) [inline]`

Create a [mln::w_window](#) from a window and a weight function.

Parameters

- [in] *win* A simple window.
- [in] *wei* A weight function.

Returns

A weighted window.

Definition at line 63 of file make/w_window.hh.

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

9.107.2.52 `template<typename W , unsigned M> mln::w_window< mln::dpoint1d, W >
mln::make::w_window1d (W(&) weights[M]) [inline]`

Create a 1D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, M, has to be a square of an odd integer.

Returns

A 1D weighted window.

Definition at line 63 of file w_window1d.hh.

References `mln::w_window< D, W >::insert()`.

9.107.2.53 `template<typename W , unsigned S> mln::w_window< mln::dpoint2d, W >
mln::make::w_window2d (W(&) weights[S]) [inline]`

Create a 2D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, S, has to be a square of an odd integer.

Returns

A 2D weighted window.

Definition at line 62 of file w_window2d.hh.

9.107.2.54 `template<typename W , unsigned M> mln::w_window< mln::dpoint3d, W >
mln::make::w_window3d (W(&) weights[M]) [inline]`

Create a 3D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, M, has to be a cube of an odd integer.

Returns

A 3D weighted window.

Definition at line 64 of file w_window3d.hh.

References `mln::w_window< D, W >::insert()`.

9.107.2.55 `template<typename D , typename W , unsigned L> mln::w_window< D, W >
mln::make::w_window_directional (const Gdpoint< D > & dp, W(&) weights[L])
[inline]`

Create a directional centered weighted window.

Parameters

- [in] *dp* A delta-point to set the orientation.
- [in] *weights* An array of weights.

Returns

A weighted window.

The window length L has to be odd.

Definition at line 61 of file `w_window_directional.hh`.

References `mln::w_window< D, W >::insert()`, and `mln::literal::zero`.

9.108 mln::math Namespace Reference

Namespace of mathematical routines.

Functions

- `template<typename T >
T abs (const T &v)`
Generic version.
- `template<unsigned n>
value::int_u< n > abs (const value::int_u< n > &v)`
Specialization for `mln::value::int_u`.
- `int abs (int v)`
Specializations for existing overloads of `std::abs`.

9.108.1 Detailed Description

Namespace of mathematical routines.

9.108.2 Function Documentation

9.108.2.1 `template<typename T > T mln::math::abs (const T & v) [inline]`

Generic version.

Definition at line 74 of file `math/abs.hh`.

Referenced by `abs()`, and `mln::morpho::line_gradient()`.

9.108.2.2 int mln::math::abs (int *v*) [inline]

Specializations for existing overloads of std::abs.

Reference: ISO/IEC 14882:2003 C++ standard, section 26.5 (C Library, [lib.c.math]).

Definition at line 79 of file math/abs.hh.

References abs().

9.108.2.3 template<unsigned n> value::int_u< n > mln::math::abs (const value::int_u< n > & *v*) [inline]

Specialization for [mln::value::int_u](#).

Definition at line 88 of file math/abs.hh.

9.109 mln::metal Namespace Reference

Namespace of meta-programming tools.

Namespaces

- namespace [impl](#)
Implementation namespace of metal namespace.
- namespace [math](#)
Namespace of static mathematical functions.

Classes

- struct [ands](#)
Ands type.
- struct [converts_to](#)
"converts-to" check.
- struct [equal](#)
Definition of a static 'equal' test.
- struct [goes_to](#)
"goes-to" check.
- struct [is](#)
"is" check.
- struct [is_a](#)
"is_a" check.

- struct [is_not](#)
"is_not" check.
- struct [is_not_a](#)
"is_not_a" static Boolean expression.

9.109.1 Detailed Description

Namespace of meta-programming tools.

9.110 mln::metal::impl Namespace Reference

Implementation namespace of metal namespace.

9.110.1 Detailed Description

Implementation namespace of metal namespace.

9.111 mln::metal::math Namespace Reference

Namespace of static mathematical functions.

Namespaces

- namespace [impl](#)
Implementation namespace of [metal::math](#) namespace.

9.111.1 Detailed Description

Namespace of static mathematical functions.

9.112 mln::metal::math::impl Namespace Reference

Implementation namespace of [metal::math](#) namespace.

9.112.1 Detailed Description

Implementation namespace of [metal::math](#) namespace.

9.113 mln::morpho Namespace Reference

Namespace of mathematical morphology routines.

Namespaces

- namespace [approx](#)
Namespace of approximate mathematical morphology routines.
- namespace [attribute](#)
Namespace of attributes used in mathematical morphology.
- namespace [elementary](#)
Namespace of image processing routines of elementary mathematical morphology.
- namespace [impl](#)
Namespace of mathematical morphology routines implementations.
- namespace [reconstruction](#)
Namespace of morphological reconstruction routines.
- namespace [tree](#)
Namespace of morphological tree-related routines.
- namespace [watershed](#)
Namespace of morphological watershed routines.

Functions

- `template<typename I >
mln::trait::concrete< I >::ret complementation (const Image< I > &input)`
- `template<typename I >
void complementation_inplace (Image< I > &input)`
- `template<typename I , typename W >
mln::trait::concrete< I >::ret contrast (const Image< I > &input, const Window< W > &win)`
- `template<typename I , typename W >
mln::trait::concrete< I >::ret dilation (const Image< I > &input, const Window< W > &win)`
Morphological dilation.
- `template<typename I , typename W >
mln::trait::concrete< I >::ret erosion (const Image< I > &input, const Window< W > &win)`
Morphological erosion.
- `template<typename I , typename W >
mln::trait::concrete< I >::ret erosion_tolerant (const Image< I > &input, const Window< W > &win, unsigned rank)`
Morphological tolerant erosion.

- `template<typename Op , typename I , typename W >`
`mln::trait::concrete< I >::ret general (const Op &op, const Image< I > &input, const Window< W > &win)`
Morphological general routine.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient (const Image< I > &input, const Window< W > &win)`
Morphological gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_external (const Image< I > &input, const Window< W > &win)`
Morphological external gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_internal (const Image< I > &input, const Window< W > &win)`
Morphological internal gradient.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_closing (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss closing of the background.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_opening (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss opening of the background.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_closing (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss closing.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_opening (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss opening.
- `template<typename I , typename W , typename O >`
`void laplacian (const Image< I > &input, const Window< W > &win, Image< O > &output)`
- `template<typename V >`
`edge_image< util::site_pair< point2d>, V, util::graph> line_gradient (const mln::image2d< V > &ima)`
Create a line graph image representing the gradient norm of a [mln::image2d](#).

- template<typename L, typename I, typename N >
mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh, L &nbasins)

Meyer's Watershed Transform (WST) algorithm.

- template<typename L, typename I, typename N >
mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh)

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- template<typename I, typename J >
mln::trait::concrete< I >::ret min (const Image< I > &lhs, const Image< J > &rhs)
- template<typename I, typename J >
void min_inplace (Image< I > &lhs, const Image< J > &rhs)
- template<typename I, typename J >
mln::trait::concrete< I >::ret minus (const Image< I > &lhs, const Image< J > &rhs)
- template<typename I, typename J >
mln::trait::concrete< I >::ret plus (const Image< I > &lhs, const Image< J > &rhs)
- template<typename I, typename W >
mln::trait::concrete< I >::ret rank_filter (const Image< I > &input, const Window< W > &win, unsigned k)

Morphological rank_filter.

- template<typename I, typename Wfg, typename Wbg >
mln::trait::concrete< I >::ret thick_miss (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)
- template<typename I, typename Wfg, typename Wbg >
mln::trait::concrete< I >::ret thickening (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)
- template<typename I, typename Wfg, typename Wbg >
mln::trait::concrete< I >::ret thin_fit (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)
- template<typename I, typename Wfg, typename Wbg >
mln::trait::concrete< I >::ret thinning (const Image< I > &input, const Window< Wfg > &win_fg, const Window< Wbg > &win_bg)

Morphological thinning.

- template<typename I, typename W >
mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Window< W > &win)

Morphological black top-hat (for background / dark objects).

- template<typename I, typename W >
mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Window< W > &win)

Morphological self-complementary top-hat.

- template<typename I, typename W >
mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Window< W > &win)

Morphological white top-hat (for object / light objects).

9.113.1 Detailed Description

Namespace of mathematical morphology routines.

9.113.2 Function Documentation

9.113.2.1 `template<typename I> mln::trait::concrete< I >::ret mln::morpho::complementation (const Image< I> & input) [inline]`

Morphological complementation: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 116 of file complementation.hh.

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

9.113.2.2 `template<typename I> void mln::morpho::complementation_inplace (Image< I> & input) [inline]`

Morphological complementation, inplace version: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 130 of file complementation.hh.

9.113.2.3 `template<typename I, typename W> mln::trait::concrete< I>::ret mln::morpho::contrast (const Image< I> & input, const Window< W> & win) [inline]`

Morphological contrast operator (based on top-hats).

This operator is $Id + wth_B - bth_B$.

Definition at line 57 of file contrast.hh.

References `plus()`, `top_hat_black()`, and `top_hat_white()`.

9.113.2.4 `template<typename I, typename W> mln::trait::concrete< I>::ret mln::morpho::dilation (const Image< I> & input, const Window< W> & win) [inline]`

Morphological dilation.

Definition at line 162 of file dilation.hh.

References `general()`.

Referenced by `gradient()`, `gradient_external()`, `hit_or_miss_background_opening()`, `hit_or_miss_opening()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.113.2.5 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::erosion (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological erosion.

Definition at line 163 of file erosion.hh.

References general().

Referenced by gradient(), gradient_internal(), laplacian(), mln::morpho::opening::approx::structural(), and mln::morpho::closing::approx::structural().

9.113.2.6 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::erosion_tolerant (const Image< I > & input, const Window< W > &
win, unsigned rank) [inline]`

Morphological tolerant erosion.

Definition at line 200 of file erosion_tolerant.hh.

9.113.2.7 `template<typename Op , typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::general (const Op & op, const Image< I > & input, const Window<
W > & win) [inline]`

Morphological general routine.

Definition at line 168 of file general.hh.

Referenced by dilation(), and erosion().

9.113.2.8 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological gradient.

This operator is $d_B - e_B$.

Definition at line 76 of file gradient.hh.

References dilation(), erosion(), minus(), and mln::test::positive().

9.113.2.9 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient_external (const Image< I > & input, const Window< W > &
win) [inline]`

Morphological external gradient.

This operator is $d_B - Id$.

Definition at line 110 of file gradient.hh.

References dilation(), minus(), and mln::test::positive().

9.113.2.10 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient_internal (const Image< I > & input, const Window< W > &
win) [inline]`

Morphological internal gradient.

This operator is $\text{Id} - e_B$.

Definition at line 93 of file `gradient.hh`.

References `erosion()`, `minus()`, and `mln::test::positive()`.

9.113.2.11 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss (const Image< I > & input, const Window< Wh > &
win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss.

This operator is $\text{HMT}_B(B_h, B_m) = e_{B_h} \wedge (e_{B_m} \circ C)$.

Definition at line 276 of file `hit_or_miss.hh`.

Referenced by `thickening()`, and `thinning()`.

9.113.2.12 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_background_closing (const Image< I > & input, const
Window< Wh > & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss closing of the background.

This operator is $C \circ \text{HMT}_{\text{TopeBG}} \circ C$.

Definition at line 357 of file `hit_or_miss.hh`.

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

9.113.2.13 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_background_opening (const Image< I > & input, const
Window< Wh > & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss opening of the background.

This operator is $\text{HMT}_{\text{TopeBG}} = \text{HMT}_{\text{Tope}}(B_m, B_h) \circ C = d_-(-B_m) \circ \text{HMT}_B(B_h, B_m)$.

Definition at line 314 of file `hit_or_miss.hh`.

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

9.113.2.14 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_closing (const Image< I > & input, const Window< Wh
> & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss closing.

This operator is $C \circ \text{HMT}_{\text{Tope}} \circ C$.

Definition at line 337 of file `hit_or_miss.hh`.

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.

9.113.2.15 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_opening (const Image< I > & input, const Window< Wh
> & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss opening.

This operator is $HMT_{Tope}(B_h, B_m) = d_{-}(-B_h) \circ HMT_{-}(B_h, B_m)$.

Definition at line 294 of file `hit_or_miss.hh`.

References `dilation()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thin_fit()`.

9.113.2.16 `template<typename I , typename W , typename O > void mln::morpho::laplacian (
const Image< I > & input, const Window< W > & win, Image< O > & output)
[inline]`

Morphological laplacian.

This operator is $(d_B - Id) - (Id - e_B)$.

Definition at line 63 of file `laplacian.hh`.

References `dilation()`, `erosion()`, `mln::data::fill()`, and `minus()`.

9.113.2.17 `template<typename V > edge_image< util::site_pair< point2d >, V, util::graph >
mln::morpho::line_gradient (const mln::image2d< V > & ima)`

Create a line graph image representing the gradient norm of a [mln::image2d](#).

Definition at line 70 of file `line_gradient.hh`.

References `mln::math::abs()`, `mln::image2d< T >::domain()`, `mln::box< P >::has()`, `mln::window< D >::insert()`, and `mln::Box< E >::nsites()`.

9.113.2.18 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret
mln::morpho::meyer_wst (const Image< I > & input, const Neighborhood< N > &
nbh, L & nbasins)`

Meyer's Watershed Transform (WST) algorithm.

Parameters

[in] *input* The input image.

[in] *nbh* The connectivity of markers.

[out] *nbasins* The number of basins.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connectivity.

Definition at line 108 of file meyer_wst.hh.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, `mln::labeling::regional_minima()`, and `mln::literal::zero`.

9.113.2.19 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret
mln::morpho::meyer_wst (const Image< I > & input, const Neighborhood< N > &
nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, *L*, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 202 of file meyer_wst.hh.

9.113.2.20 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::min
(const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological min: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 112 of file morpho/min.hh.

9.113.2.21 `template<typename I , typename J > void mln::morpho::min_inplace (Image< I > &
lhs, const Image< J > & rhs) [inline]`

Morphological min, inplace version: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 125 of file morpho/min.hh.

9.113.2.22 `template<typename I , typename J > mln::trait::concrete< I >::ret
mln::morpho::minus (const Image< I > & lhs, const Image< J > & rhs)
[inline]`

Morphological minus: either a logical "and not" (if morpho on sets) or an arithmetical minus (if morpho on functions).

Definition at line 87 of file morpho/minus.hh.

Referenced by `gradient()`, `gradient_external()`, `gradient_internal()`, `laplacian()`, `thin_fit()`, `thinning()`, `top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `top_hat_self_complementary()`, `mln::morpho::elementary::top_hat_self_complementary()`, `top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

9.113.2.23 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::plus (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological plus: either a "logical or" (if morpho on sets) or an "arithmetical plus" (if morpho on functions).

Definition at line 86 of file morpho/plus.hh.

Referenced by `contrast()`, `thick_miss()`, and `thickening()`.

9.113.2.24 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::rank_filter (const Image< I > & input, const Window< W > & win, unsigned k) [inline]`

Morphological `rank_filter`.

Definition at line 213 of file rank_filter.hh.

9.113.2.25 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thick_miss (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thick-miss.

This operator is $THICK_B = Id + HMT_{TopeBG_B}$, where $B = (Bfg, Bbg)$.

Definition at line 59 of file thick_miss.hh.

References `hit_or_miss_background_opening()`, and `plus()`.

9.113.2.26 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thickening (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thickening.

This operator is $THICK_B = Id + HMT_B$, where $B = (Bfg, Bbg)$.

Definition at line 88 of file thickening.hh.

References `hit_or_miss()`, and `plus()`.

Referenced by `thinning()`.

9.113.2.27 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thin_fit (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thin-fit.

This operator is $THIN_B = Id - HMT_{Tope_B}$ where $B = (Bfg, Bbg)$.

Definition at line 87 of file thin_fit.hh.

References `hit_or_miss_opening()`, and `minus()`.

9.113.2.28 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thinning (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thinning.

This operator is $THIN_B = Id - HMT_B$, where $B = (Bfg, Bbg)$.

Definition at line 90 of file `thinning.hh`.

References `complementation()`, `hit_or_miss()`, `minus()`, and `thickening()`.

9.113.2.29 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_black (const Image< I > & input, const Window< W > & win) [inline]`

Morphological black top-hat (for background / dark objects).

This operator is $clo_B - Id$.

Definition at line 102 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

9.113.2.30 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_self_complementary (const Image< I > & input, const Window< W > & win) [inline]`

Morphological self-complementary top-hat.

This operator is

$= top_hat_white + top_hat_black$

$= (input - opening) + (closing - input)$

$= closing - opening$.

Definition at line 121 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

9.113.2.31 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_white (const Image< I > & input, const Window< W > & win) [inline]`

Morphological white top-hat (for object / light objects).

This operator is $Id - ope_B$.

Definition at line 83 of file `top_hat.hh`.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

9.114 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

9.114.1 Detailed Description

Namespace of approximate mathematical morphology routines.

9.115 mln::morpho::attribute Namespace Reference

Namespace of attributes used in mathematical morphology.

Classes

- class [card](#)
Cardinality accumulator class.
- struct [count_adjacent_vertices](#)
Count_Adjacent_Vertices accumulator class.
- struct [height](#)
Height accumulator class.
- struct [sharpness](#)
Sharpness accumulator class.
- class [sum](#)
Suminality accumulator class.
- struct [volume](#)
Volume accumulator class.

9.115.1 Detailed Description

Namespace of attributes used in mathematical morphology.

9.116 mln::morpho::closing::approx Namespace Reference

Namespace of approximate mathematical morphology closing routines.

Functions

- template<typename I, typename W >
mln::trait::concrete< I >::ret [structural](#) (const [Image](#)< I > &input, const [Window](#)< W > &win)
Approximate of morphological structural closing.

9.116.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

9.116.2 Function Documentation

9.116.2.1 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::closing::approx::structural (const Image< I > & input, const
Window< W > & win) [inline]`

Approximate of morphological structural closing.

This operator is $e_{\{-B\}} \circ d_B$.

Definition at line 65 of file closing/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.117 mln::morpho::elementary Namespace Reference

Namespace of image processing routines of elementary mathematical morphology.

Functions

- `template<typename I , typename N >
mln::trait::concrete< I >::ret closing (const Image< I > &input, const Neighborhood< N >
&nbh)`
Morphological elementary closing.
- `template<typename I , typename N >
mln_trait_op_minus_twice (typename mln::trait::concrete< I >::ret) laplacian(const Image< I >
&input)`
Morphological elementary laplacian.
- `template<typename I , typename N >
mln::trait::concrete< I >::ret opening (const Image< I > &input, const Neighborhood< N >
&nbh)`
Morphological elementary opening.
- `template<typename I , typename N >
mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Neighborhood< N >
&nbh)`
Morphological elementary black top-hat (for background / dark objects).
- `template<typename I , typename N >
mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Neigh-
borhood< N > &nbh)`
Morphological elementary self-complementary top-hat.

- `template<typename I , typename N > mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary white top-hat (for object / light objects).

9.117.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

9.117.2 Function Documentation

9.117.2.1 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::closing (const Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary closing.

This operator is $e \circ d$.

Definition at line 58 of file closing.hh.

Referenced by `top_hat_black()`, and `top_hat_self_complementary()`.

9.117.2.2 `template<typename I , typename N > mln::morpho::elementary::mln_trait_op_minus_twice (typename mln::trait::concrete< I >::ret) const [inline]`

Morphological elementary laplacian.

This operator is $(d - id) - (id - e)$.

9.117.2.3 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::opening (const Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary opening.

This operator is $d \circ e$.

Definition at line 58 of file opening.hh.

Referenced by `top_hat_self_complementary()`, and `top_hat_white()`.

9.117.2.4 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::top_hat_black (const Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary black top-hat (for background / dark objects).

This operator is $clo - Id$.

Definition at line 105 of file elementary/top_hat.hh.

References `closing()`, `mln::morpho::minus()`, and `mln::test::positive()`.

9.117.2.5 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::top_hat_self_complementary (const Image< I > & input,
const Neighborhood< N > & nbh) [inline]`

Morphological elementary self-complementary top-hat.

This operator is

= top_hat_white + top_hat_black

= (Id - opening) + (closing - Id)

= closing - opening.

Definition at line 125 of file elementary/top_hat.hh.

References closing(), mln::morpho::minus(), opening(), and mln::test::positive().

9.117.2.6 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::top_hat_white (const Image< I > & input, const
Neighborhood< N > & nbh) [inline]`

Morphological elementary white top-hat (for object / light objects).

This operator is Id - ope.

Definition at line 85 of file elementary/top_hat.hh.

References mln::morpho::minus(), opening(), and mln::test::positive().

9.118 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

Namespaces

- namespace [generic](#)

Namespace of mathematical morphology routines generic implementations.

9.118.1 Detailed Description

Namespace of mathematical morphology routines implementations.

9.119 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines generic implementations.

9.119.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

9.120 mln::morpho::opening::approx Namespace Reference

Namespace of approximate mathematical morphology opening routines.

Functions

- `template<typename I , typename W > mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`
Approximate of morphological structural opening.

9.120.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

9.120.2 Function Documentation

- 9.120.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::opening::approx::structural (const Image< I > & input, const
Window< W > & win) [inline]`

Approximate of morphological structural opening.

This operator is $d_{\{-B\}} \circ e_B$.

Definition at line 64 of file opening/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.121 mln::morpho::reconstruction Namespace Reference

Namespace of morphological reconstruction routines.

Namespaces

- namespace `by_dilation`
Namespace of morphological reconstruction by dilation routines.
- namespace `by_erosion`
Namespace of morphological reconstruction by erosion routines.

9.121.1 Detailed Description

Namespace of morphological reconstruction routines.

9.122 mln::morpho::reconstruction::by_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

9.122.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

9.123 mln::morpho::reconstruction::by_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

9.123.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

9.124 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

Namespaces

- namespace [filter](#)
Namespace for attribute filtering.

Functions

- template<typename A , typename T >
mln::trait::ch_value< typename T::function, typename A::result >::ret [compute_attribute_image](#)
(const [Accumulator](#)< A > &a, const T &t, mln::trait::ch_value< typename T::function, A >::ret *accu_image=0)
Compute an attribute image using tree with a parent relationship between sites.
- template<typename A , typename T , typename V >
mln::trait::ch_value< typename T::function, typename A::result >::ret [compute_attribute_image_from](#)
(const [Accumulator](#)< A > &a, const T &t, const [Image](#)< V > &values, mln::trait::ch_value< typename T::function, A >::ret *accu_image=0)
The same as compute_attribute_image but uses the values stored by values image instead.
- template<typename I , typename N , typename S >
mln::trait::ch_value< I , typename I::psite >::ret [compute_parent](#) (const [Image](#)< I > &f, const [Neighborhood](#)< N > &nbh, const [Site_Set](#)< S > &s)

Compute a tree with a parent relationship between sites.

- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > dual_input_max_tree (const Image< I > &f, const Image<`
`I > &m, const Neighborhood< N > &nbh)`

Compute the dual input max tree using mask-based connectivity.

- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > max_tree (const Image< I > &f, const Neighborhood< N`
`> &nbh)`

Compute a canonized max-tree.

- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > min_tree (const Image< I > &f, const Neighborhood< N`
`> &nbh)`

Compute a canonized min-tree.

- `template<typename T , typename A , typename P , typename W >`
`void propagate_if (const T &tree, Image< A > &a_, const way_of_propagation< W > &prop_,`
`const Function_v2b< P > &pred_, const typename A::value &v)`
- `template<typename T , typename A , typename P >`
`void propagate_if (const T &tree, Image< A > &a_, const desc_propagation &prop_, const`
`Function_v2b< P > &pred_)`
- `template<typename T , typename A , typename W >`
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`
`&prop_, const typename A::value &v)`
- `template<typename T , typename A , typename W >`
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`
`&prop_, const typename A::value &v, const typename A::value &v_prop)`
- `template<typename T , typename A >`
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_, const type-`
`name A::value &v)`
- `template<typename T , typename A >`
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_)`
- `template<typename T , typename A >`
`void propagate_node_to_descendants (typename A::psite n, const T &t, Image< A > &a_, const`
`typename A::value &v, unsigned *nb_leaves=0)`
- `template<typename T , typename A >`
`void propagate_node_to_descendants (typename A::psite &n, const T &t, Image< A > &a_, un-`
`signed *nb_leaves=0)`
- `template<typename T , typename F >`
`void propagate_representative (const T &t, Image< F > &f_)`

Propagate the representative node's value to non-representative points of the component.

9.124.1 Detailed Description

Namespace of morphological tree-related routines.

9.124.2 Function Documentation

9.124.2.1 `template<typename A , typename T > mln::trait::ch_value< typename T::function, typename A::result >::ret mln::morpho::tree::compute_attribute_image (const Accumulator< A > & a, const T & t, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0) [inline]`

Compute an attribute image using tree with a parent relationship between sites.

In the attribute image, the resulting value at a node is the 'sum' of its sub-components value + the attribute value at this node.

Warning: *s* translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to *s*.

It is very convenient since all processing upon the parent tree are performed following *s* (in the default "forward" way).

FIXME: Put it more clearly...

The parent result image verifies:

- *p* is root iff `parent(p) == p`
- *p* is a node iff either *p* is root or `f(parent(p)) != f(p)`.

Parameters

[in] *a* Attribute.

[in] *t* Component tree.

[out] *accu_image* Optional argument used to store image of attribute accumulator.

Returns

The attribute image.

Definition at line 216 of file `compute_attribute_image.hh`.

Referenced by `compute_attribute_image_from()`.

9.124.2.2 `template<typename A , typename T , typename V > mln::trait::ch_value< typename T::function, typename A::result >::ret mln::morpho::tree::compute_attribute_image_from (const Accumulator< A > & a, const T & t, const Image< V > & values, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0) [inline]`

The same as `compute_attribute_image` but uses the values stored by *values* image instead.

Parameters

[in] *a* Attribute.

[in] *t* Component tree.

[in] *values* Value image.

[out] *accu_image* Optional argument used to store image.

Returns

Definition at line 233 of file compute_attribute_image.hh.

References compute_attribute_image().

9.124.2.3 `template<typename I , typename N , typename S > mln::trait::ch_value< I, typename I::psite >::ret mln::morpho::tree::compute_parent (const Image< I > & f, const Neighborhood< N > & nbh, const Site_Set< S > & s) [inline]`

Compute a tree with a parent relationship between sites.

Warning: s translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to s .

It is very convenient since most processing routines upon the parent tree are performed following s (in the default "forward" way). Indeed that is the way to propagate information from parents to children.

The parent result image verifies:

- p is root iff $\text{parent}(p) == p$
- p is a node iff either p is root or $f(\text{parent}(p)) \neq f(p)$.

The choice " s means childhood" is consistent with labeling in binary images. In that particular case, while browsing the image in forward scan (video), we expect to find first a tree root (a first point, representative of a component) and then the other component points. Please note that it leads to increasing values of labels in the "natural" video scan.

Since mathematical morphology on functions is related to morphology on sets, we clearly want to keep the equivalence between "component labeling" and "component filtering" using trees.

FIXME: Put it more clearly... Insert pictures!

A binary image:

- $\begin{vmatrix} | & | \\ - & - \end{vmatrix}$
- $\begin{vmatrix} | & | & - \\ | & - & | \end{vmatrix}$
- $\begin{vmatrix} - & - & - & - \end{vmatrix}$
- $\begin{vmatrix} - & | & | & - \end{vmatrix}$

where ' $|$ ' means true and ' $-$ ' means false.

Its labeling:

```
0 1 1 0 0
0 1 1 0 2
0 0 0 0 0
0 0 3 3 0
```

The corresponding forest:

```
x o . x x
x . . x o
x x x x x
x x o . x
```

where 'x' means "no data", 'o' is a tree root (representative point for a component), and '.' is a tree regular (non-root) point (in a component by not its representative point).

The forest, with the parent relationship looks like:

```
o < .
^ r
. . o
o < .
```

Definition at line 286 of file compute_parent.hh.

9.124.2.4 `template<typename I, typename N> morpho::tree::data< I, p_array< typename I::psite >> mln::morpho::tree::dual_input_max_tree (const Image< I> & f, const Image< I> & m, const Neighborhood< N> & nbh) [inline]`

Compute the dual input max tree using mask-based connectivity.

Parameters

[in] *f* The original image.
[in] *m* The connectivity mask.
[in] *nbh* The neighborhood of the mask.

Returns

The computed tree.

Definition at line 109 of file dual_input_tree.hh.

9.124.2.5 `template<typename I, typename N> data< I, p_array< typename I::psite >> mln::morpho::tree::max_tree (const Image< I> & f, const Neighborhood< N> & nbh) [inline]`

Compute a canonized max-tree.

Parameters

[in] *f* The input image.
[in] *nbh* The neighborhood.

Returns

The corresponding max-tree structure.

Definition at line 100 of file component_tree.hh.

References `mln::data::sort_psites_increasing()`.

9.124.2.6 `template<typename I, typename N> data< I, p_array< typename I::psite >> mln::morpho::tree::min_tree (const Image< I> & f, const Neighborhood< N> & nbh) [inline]`

Compute a canonized min-tree.

Parameters

- [in] *f* The input image.
- [in] *nbh* The neighborhood.

Returns

The corresponding min-tree structure.

Definition at line 77 of file component_tree.hh.

References mln::data::sort_psites_decreasing().

9.124.2.7 `template<typename T , typename A , typename P , typename W > void
mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop_, const Function_v2b< P > & pred_, const
typename A::value & v) [inline]`

Propagate nodes checking the predicate *pred* in the way defined by *way_of_propagation*.

Parameters

- tree* Component tree used for propagation.
- a_* Attributed image where values are propagated.
- prop_* Propagate node in ascendant or descendant way.
- pred_* Predicate that node must check to be propagated.
- v* [Value](#) to be propagated. (By default *v* is the value at the node being propagated).

Definition at line 293 of file propagate_if.hh.

Referenced by propagate_if(), propagate_if_value(), and mln::morpho::tree::filter::subtractive().

9.124.2.8 `template<typename T , typename A , typename P > void
mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const
desc_propagation & prop_, const Function_v2b< P > & pred_) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 310 of file propagate_if.hh.

References propagate_if().

9.124.2.9 `template<typename T , typename A , typename W > void
mln::morpho::tree::propagate_if_value (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop_, const typename A::value & v) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 278 of file propagate_if.hh.

References propagate_if().

9.124.2.10 `template<typename T , typename A , typename W > void
mln::morpho::tree::propagate_if_value (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop_, const typename A::value & v, const typename
A::value & v_prop) [inline]`

Propagate nodes having the value *v* in the way defined by *way_of_propagation*.

Parameters

tree Component tree used for propagation.

a_ Attributed image where values are propagated.

prop_ Propagate node in ascendant or descendant way.

v [Value](#) that node must have to be propagated.

v_prop [Value](#) to propagate (By default it is the value at the node being propagated).

Definition at line 262 of file `propagate_if.hh`.

References `propagate_if()`.

9.124.2.11 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors (typename A::psite n, const T & t, Image< A > & a_, const typename
A::value & v)`

Propagate a value *v* from a node *n* to its ancestors.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in] *a_* Attribute image where values are propagated.

[in] *v* [Value](#) to propagate.

Definition at line 170 of file `propagate_node.hh`.

Referenced by `propagate_node_to_ancestors()`.

9.124.2.12 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors (typename A::psite n, const T & t, Image< A > & a_)
[inline]`

Propagate the node's value to its ancestors.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in, out] *a_* Attribute image where values are propagated.

Definition at line 197 of file `propagate_node.hh`.

References `propagate_node_to_ancestors()`.

9.124.2.13 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants (typename A::psite n, const T & t, Image< A > & a_, const typename A::value & v, unsigned * nb_leaves = 0) [inline]`

Propagate a value *v* from a node *n* to its descendants.

Parameters

- [in] *n* Node to propagate.
- [in] *t* Component tree used for propagation.
- [in] *a_* Attribute image where values are propagated.
- [in] *v* Value to propagate.
- [out] *nb_leaves* Optional. Store the number of leaves in the component.

Definition at line 120 of file propagate_node.hh.

9.124.2.14 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants (typename A::psite & n, const T & t, Image< A > & a_, unsigned * nb_leaves = 0) [inline]`

Propagate the node's value to its descendants.

Parameters

- [in] *n* Node to propagate.
- [in] *t* Component tree used for propagation.
- [in] *a_* Attribute image where values are propagated.
- [out] *nb_leaves* Optional. Store the number of leaves in the component.

9.124.2.15 `template<typename T , typename F > void mln::morpho::tree::propagate_representative (const T & t, Image< F > & f_) [inline]`

Propagate the representative node's value to non-representative points of the component.

Parameters

- t* Component tree.
- f_* Value image.

Definition at line 65 of file propagate_representative.hh.

9.125 mln::morpho::tree::filter Namespace Reference

Namespace for attribute filtering.

Functions

- `template<typename T , typename F , typename P >`
`void direct (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Direct non-pruning strategy.
- `template<typename T , typename F , typename P >`
`void filter (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_, const typename F::value &v)`
Filter the image f_ with a given value.
- `template<typename T , typename F , typename P >`
`void max (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Max pruning strategy.
- `template<typename T , typename F , typename P >`
`void min (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Min pruning strategy.
- `template<typename T , typename F , typename P >`
`void subtractive (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Subtractive pruning strategy.

9.125.1 Detailed Description

Namespace for attribute filtering.

9.125.2 Function Documentation

9.125.2.1 `template<typename T , typename F , typename P > void`
`mln::morpho::tree::filter::direct (const T & tree, Image< F > & f_, const`
`Function_v2b< P > & pred_) [inline]`

Direct non-pruning strategy.

A node is removed if it does not verify the predicate. The sub-components remain intact.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 73 of file direct.hh.

9.125.2.2 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::filter`
`(const T & tree, Image< F > & f_, const Function_v2b< P > & pred_, const`
`typename F::value & v) [inline]`

Filter the image f_ with a given value.

The sub-components of nodes that does not match the predicate `pred_` are filled with the given value `v`.

Parameters

- tree* Component tree.
- f_* [Image](#) function.
- pred_* Predicate.
- v* [Value](#) to propagate.

Definition at line 73 of file `morpho/tree/filter/filter.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.3 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::max (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Max pruning strategy.

A node is removed iff all of its children are removed or if it does not verify the predicate `pred_`.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 74 of file `morpho/tree/filter/max.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.4 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::min (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Min pruning strategy.

A node is removed iff its parent is removed or if it does not verify the predicate `pred_`.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 75 of file `morpho/tree/filter/min.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.5 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::subtractive (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Subtractive pruning strategy.

The node is removed if it does not verify the predicate. The sub-components values are set to the value of the removed component.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 77 of file subtractive.hh.

References `mln::morpho::tree::propagate_if()`.

9.126 mln::morpho::watershed Namespace Reference

Namespace of morphological watershed routines.

Namespaces

- namespace [watershed](#)
Namespace of morphological watershed routines implementations.

Functions

- `template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N > &nbh, L &n_basins)`
Meyer's Watershed Transform (WST) algorithm.
- `template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N > &nbh)`
Meyer's Watershed Transform (WST) algorithm, with no count of basins.
- `template<typename I , typename J >
mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &ws_ima)`
Convert an image to a rgb8 image and draw the watershed lines.
- `template<typename I , typename J >
mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &ws_ima_, const value::rgb8 &wsl_color)`
Convert an image to a rgb8 image and draw the watershed lines.
- `template<class T >
T::image_t topological (T &tree)`
Compute a toological watershed transform from tree.

9.126.1 Detailed Description

Namespace of morphological watershed routines.

9.126.2 Function Documentation

9.126.2.1 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret
mln::morpho::watershed::flooding (const Image< I > & input, const Neighborhood<
N > & nbh, L & n_basins) [inline]`

Meyer's Watershed Transform (WST) algorithm.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

[out] *n_basins* The number of basins.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connexity.

Definition at line 381 of file flooding.hh.

9.126.2.2 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret
mln::morpho::watershed::flooding (const Image< I > & input, const Neighborhood<
N > & nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, *L*, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 396 of file flooding.hh.

9.126.2.3 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::morpho::watershed::superpose (const Image< I > & input, const Image< J > &
ws_ima) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 109 of file morpho/watershed/superpose.hh.

References `mln::literal::red`, and `superpose()`.

9.126.2.4 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::morpho::watershed::superpose (const Image< I > & input_, const Image< J >
& ws_ima_, const value::rgb8 & wsl_color) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 85 of file morpho/watershed/superpose.hh.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

9.126.2.5 `template<class T > T::image_t mln::morpho::watershed::topological (T & tree)`

Compute a toological watershed transform from *tree*.

Definition at line 675 of file topological.hh.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, and `topological()`.

Referenced by `topological()`.

9.127 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological watershed routines implementations.

Namespaces

- namespace [generic](#)

Namespace of morphological watershed routines generic implementations.

9.127.1 Detailed Description

Namespace of morphological watershed routines implementations.

9.128 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological watershed routines generic implementations.

9.128.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

9.129 mln::norm Namespace Reference

Namespace of norms.

Namespaces

- namespace [impl](#)
Implementation namespace of norm namespace.

Functions

- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2 (const C(&vec)[n])`
L2-norm of a vector vec.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1 (const C(&vec)[n])`
L1-norm of a vector vec.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1_distance (const C(&vec1)[n], const C(&vec2)[n])`
L1-norm distance between vectors vec1 and vec2.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum sqr_l2 (const C(&vec)[n])`
Squared L2-norm of a vector vec.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2_distance (const C(&vec1)[n], const C(&vec2)[n])`
L2-norm distance between vectors vec1 and vec2.
- `template<unsigned n, typename C >`
`C linfty (const C(&vec)[n])`
L-infinity-norm of a vector vec.
- `template<unsigned n, typename C >`
`C linfty_distance (const C(&vec1)[n], const C(&vec2)[n])`
L-infinity-norm distance between vectors vec1 and vec2.

9.129.1 Detailed Description

Namespace of norms.

9.129.2 Function Documentation

9.129.2.1 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l1 (const C(&) vec[n])
[inline]`

L1-norm of a vector *vec*.

Definition at line 108 of file l1.hh.

9.129.2.2 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l1_distance (const C(&)
vec1[n], const C(&) vec2[n]) [inline]`

L1-norm distance between vectors *vec1* and *vec2*.

Definition at line 124 of file l1.hh.

9.129.2.3 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2 (const C(&) vec[n])
[inline]`

L2-norm of a vector *vec*.

Definition at line 139 of file l2.hh.

9.129.2.4 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2_distance (const C(&)
vec1[n], const C(&) vec2[n]) [inline]`

L2-norm distance between vectors *vec1* and *vec2*.

Definition at line 173 of file l2.hh.

9.129.2.5 `template<unsigned n, typename C > C mln::norm::linfty (const C(&) vec[n])
[inline]`

L-infinity-norm of a vector *vec*.

Definition at line 113 of file linfty.hh.

9.129.2.6 `template<unsigned n, typename C > C mln::norm::linfty_distance (const C(&)
vec1[n], const C(&) vec2[n]) [inline]`

L-infinity-norm distance between vectors *vec1* and *vec2*.

Definition at line 127 of file linfty.hh.

9.129.2.7 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::sqr_l2 (const C(&) vec[n])
[inline]`

Squared L2-norm of a vector *vec*.

Definition at line 156 of file l2.hh.

Referenced by `mln::geom::mesh_corner_point_area()`, and `mln::geom::mesh_normal()`.

9.130 mln::norm::impl Namespace Reference

Implementation namespace of norm namespace.

9.130.1 Detailed Description

Implementation namespace of norm namespace.

9.131 mln::opt Namespace Reference

Namespace of optional routines.

Namespaces

- namespace [impl](#)
Implementation namespace of opt namespace.

Functions

- `template<typename I >
I::rvalue at (const Image< I > &ima, def::coord ind)
One dimension Read-only access to the ima value located at (ind).`
- `template<typename I >
I::lvalue at (Image< I > &ima, def::coord ind)
Read-write access to the ima value located at (ind).`
- `template<typename I >
I::lvalue at (Image< I > &ima, def::coord row, def::coord col)
Read-write access to the ima value located at (row, col).`
- `template<typename I >
I::rvalue at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)
Three dimensions Read-only access to the ima value located at (sli, row, col).`
- `template<typename I >
I::rvalue at (const Image< I > &ima, def::coord row, def::coord col)`

Two dimensions Read-only access to the `ima` value located at `(row, col)`.

- `template<typename I>`
`I::lvalue at (Image< I> &ima, def::coord sli, def::coord row, def::coord col)`

Read-write access to the `ima` value located at `(sli, row, col)`.

9.131.1 Detailed Description

Namespace of optional routines.

9.131.2 Function Documentation

9.131.2.1 `template<typename I> I::rvalue mln::opt::at (const Image< I> & ima, def::coord ind) [inline]`

One dimension Read-only access to the `ima` value located at `(ind)`.

Definition at line 151 of file `at.hh`.

Referenced by `mln::transform::hough()`, and `mln::make::image()`.

9.131.2.2 `template<typename I> I::lvalue mln::opt::at (Image< I> & ima, def::coord ind)`

Read-write access to the `ima` value located at `(ind)`.

Definition at line 160 of file `at.hh`.

9.131.2.3 `template<typename I> I::lvalue mln::opt::at (Image< I> & ima, def::coord row, def::coord col)`

Read-write access to the `ima` value located at `(row, col)`.

Definition at line 245 of file `at.hh`.

9.131.2.4 `template<typename I> I::rvalue mln::opt::at (const Image< I> & ima, def::coord sli, def::coord row, def::coord col) [inline]`

Three dimensions Read-only access to the `ima` value located at `(sli, row, col)`.

Definition at line 320 of file `at.hh`.

9.131.2.5 `template<typename I> I::rvalue mln::opt::at (const Image< I> & ima, def::coord row, def::coord col) [inline]`

Two dimensions Read-only access to the `ima` value located at `(row, col)`.

Definition at line 236 of file `at.hh`.

9.131.2.6 `template<typename I> I::lvalue mln::opt::at (Image< I> & ima, def::coord sli, def::coord row, def::coord col)`

Read-write access to the `ima` value located at `(sli, row, col)`.

Definition at line 330 of file `at.hh`.

9.132 mln::opt::impl Namespace Reference

Implementation namespace of `opt` namespace.

9.132.1 Detailed Description

Implementation namespace of `opt` namespace. Three dimensions.

Two dimensions.

One dimension.

9.133 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [image](#)
A generic point-wise image implementation.

9.133.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134 mln::registration Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [closest_point_basic](#)
Closest point functor based on map distance.
- class [closest_point_with_map](#)
Closest point functor based on map distance.

Functions

- `template<typename P , typename F >`
`algebra::quat get_rot (const p_array< P > &P_, const vec3d_f &mu_P, const vec3d_f &mu_Yk,`
`const F &closest_point, const algebra::quat &qR, const vec3d_f &qT)`

FIXME: work only for 3d images.

- `template<typename P , typename F >`
`std::pair< algebra::quat, mln_vec(P)> icp (const p_array< P > &P_, const p_array< P > &X, const`
`F &closest_point, const algebra::quat &initial_rot, const mln_vec(P)&initial_translation)`

Base version of the ICP algorithm. It is called in other variants.

- `template<typename P , typename F >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > icp (const p_array< P > &P_,`
`const p_array< P > &X, const F &closest_point)`

- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration1 (const box< P`
`> &domain, const p_array< P > &P_, const p_array< P > &X)`

Call ICP once and return the resulting transformation.

- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration2 (const box< P`
`> &domain, const p_array< P > &P_, const p_array< P > &X)`

Call ICP 10 times.

- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration3 (const box< P`
`> &domain, const p_array< P > &P_, const p_array< P > &X)`

Call ICP 10 times.

9.134.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134.2 Function Documentation

- 9.134.2.1** `template<typename P , typename F > algebra::quat mln::registration::get_rot (const`
`p_array< P > & P_, const vec3d_f & mu_P, const vec3d_f & mu_Yk, const F &`
`closest_point, const algebra::quat & qR, const vec3d_f & qT)`

FIXME: work only for 3d images.

Definition at line 527 of file icp.hh.

References `mln::p_array< P >::nsites()`.

9.134.2.2 `template<typename P , typename F > std::pair< algebra::quat, mln_vec(P)>
 mln::registration::icp (const p_array< P > & P_, const p_array< P > & X, const F &
 closest_point, const algebra::quat & initial_rot, const mln_vec(P)& initial_translation)
 [inline]`

Base version of the ICP algorithm. It is called in other variants.

Register point in `c` using a function of closest points `closest_point`. This overload allows to specify initial transformations.

Parameters

- [in] *P_* The cloud of points.
- [in] *X* the reference surface.
- [in] *closest_point* The function of closest points.
- [in] *initial_rot* An initial rotation.
- [in] *initial_translation* An initial translation.

Returns

the rigid transformation which may be use later to create a registered image.

WARNING: the function `closest_point` *MUST* take float/double vector as arguments. Otherwise the resulting transformation may be wrong due to the truncation of the vector coordinate values.

Precondition

`P_` and `X` must not be empty.

Reference article: "A Method for Registration of 3-D Shapes", Paul J. Besl and Neil D. McKay, IEEE, 2, February 1992.

Definition at line 612 of file `icp.hh`.

References `mln::geom::bbox()`, `mln::literal::black`, `mln::set::compute()`, `mln::duplicate()`, `mln::box< P >::enlarge()`, `mln::data::fill()`, `mln::literal::green`, `mln::io::ppm::save()`, and `mln::literal::white`.

9.134.2.3 `template<typename P , typename F > composed<
 translation<P::dim,float>,rotation<P::dim,float> > mln::registration::icp (const
 p_array< P > & P_, const p_array< P > & X, const F & closest_point)`

Register point in `c` using a function of closest points `closest_point`.

Parameters

- [in] *P_* The cloud of points.
- [in] *X* the reference surface.
- [in] *closest_point* The function of closest points.

Returns

the rigid transformation which may be use later to create a registered image.

9.134.2.4 `template<typename P> composed< translation< P::dim, float >, rotation< P::dim, float >> mln::registration::registration1 (const box< P> & domain, const p_array< P> & P_, const p_array< P> & X) [inline]`

Call ICP once and return the resulting transformation.

Definition at line 325 of file registration.hh.

9.134.2.5 `template<typename P> composed< translation< P::dim, float >, rotation< P::dim, float >> mln::registration::registration2 (const box< P> & domain, const p_array< P> & P_, const p_array< P> & X) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset of which size is decreasing. For each call, a distance criterion is computed on a subset. Sites part of the subset which are too far or too close are removed. Removed sites are **NOT** reused later in the subset.

Definition at line 345 of file registration.hh.

9.134.2.6 `template<typename P> composed< translation< P::dim, float >, rotation< P::dim, float >> mln::registration::registration3 (const box< P> & domain, const p_array< P> & P_, const p_array< P> & X) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset. For each call, a distance criterion is computed on a subset. A new subset is computed from the whole set of points according to this distance. It will be used in the next call. Removed Sites **MAY** be reintegrated.

Definition at line 365 of file registration.hh.

9.135 mln::select Namespace Reference

Select namespace (FIXME doc).

Classes

- struct [p_of](#)
Structure [p_of](#).

9.135.1 Detailed Description

Select namespace (FIXME doc).

9.136 mln::set Namespace Reference

Namespace of image processing routines related to pixel sets.

Functions

- `template<typename S >`
`unsigned card (const Site_Set< S > &s)`
Compute the cardinality of the site set s .
- `template<typename A , typename S >`
`A::result compute (const Accumulator< A > &a, const Site_Set< S > &s)`
Compute an accumulator onto a site set.
- `template<typename A , typename I >`
`A::result compute_with_weights (const Accumulator< A > &a, const Image< I > &w)`
Compute an accumulator on a site set described by an image.
- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute_with_weights (const Accumulator< A > &a, const Image< I > &w, const Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator on every labeled sub-site-sets.
- `template<typename S >`
`S::site get (const Site_Set< S > &s, size_t index)`
FIXME.
- `template<typename S >`
`bool has (const Site_Set< S > &s, const typename S::site &e)`
FIXME.
- `template<typename A , typename S >`
`mln_meta_accu_result (A, typename S::site) compute(const Meta_Accumulator< A > &a)`
Compute an accumulator onto a site set.
- `template<typename A , typename I >`
`mln_meta_accu_result (A, typename I::site) compute_with_weights(const Meta_Accumulator< A > &a)`
Compute an accumulator on a site set described by an image.

9.136.1 Detailed Description

Namespace of image processing routines related to pixel sets.

9.136.2 Function Documentation

9.136.2.1 `template<typename S > unsigned mln::set::card (const Site_Set< S > & s)` `[inline]`

Compute the cardinality of the site set s .

Definition at line 134 of file set/card.hh.

9.136.2.2 `template<typename A , typename S > A::result mln::set::compute (const Accumulator< A > & a, const Site_Set< S > & s) [inline]`

Compute an accumulator onto a site set.

Parameters

[in] *a* An accumulator.

[in] *s* A site set.

Returns

The accumulator result.

Definition at line 112 of file set/compute.hh.

Referenced by mln::registration::icp().

9.136.2.3 `template<typename A , typename I > A::result mln::set::compute_with_weights (const Accumulator< A > & a, const Image< I > & w) [inline]`

Compute an accumulator on a site set described by an image.

Parameters

[in] *a* An accumulator.

[in] *w* An image of weights (a site -> a weight).

Returns

The accumulator result.

Definition at line 217 of file compute_with_weights.hh.

Referenced by compute_with_weights().

9.136.2.4 `template<typename A , typename I , typename L > util::array< typename A::result > mln::set::compute_with_weights (const Accumulator< A > & a, const Image< I > & w, const Image< L > & label, const typename L::value & nlabels)`

Compute an accumulator on every labeled sub-site-sets.

Parameters

[in] *a* An accumulator.

[in] *w* An image of weights (a site -> a weight).

[in] *label* A label image.

[in] *nlabels* The number of labels in *label*.

Returns

An array of accumulator result. One per label.

Definition at line 234 of file compute_with_weights.hh.

References compute_with_weights().

9.136.2.5 `template<typename S> S::site mln::set::get (const Site_Set< S> & s, size_t index)`

FIXME.

Definition at line 56 of file set/get.hh.

9.136.2.6 `template<typename S> bool mln::set::has (const Site_Set< S> & s, const typename S::site & e)`

FIXME.

Definition at line 56 of file set/has.hh.

9.136.2.7 `template<typename A, typename S> mln::set::mln_meta_accu_result (A, typename S::site) const`

Compute an accumulator onto a site set.

Parameters

[in] *a* A meta-accumulator.

[in] *s* A site set.

9.136.2.8 `template<typename A, typename I> mln::set::mln_meta_accu_result (A, typename I::site) const [inline]`

Compute an accumulator on a site set described by an image.

Parameters

[in] *a* A meta-accumulator.

[in] *w* An image of weights (a site -> a weight).

Returns

The accumulator result.

9.137 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I>
mln::trait::concrete< I>::ret antialiased (const Image< I> &input, unsigned factor, const typename I::domain_t &output_domain, unsigned border_thickness)`
Antialiased subsampling.
- `template<typename I>
mln::trait::concrete< I>::ret antialiased (const Image< I> &input, unsigned factor)`

- `template<typename I>`
`mln::trait::concrete< I >::ret gaussian_subsampling (const Image< I > &input, float sigma, const`
`typename I::dpsite &first_p, const typename I::site::coord &gap)`
Gaussian subsampling FIXME : doxy.
- `template<typename I>`
`mln::trait::concrete< I >::ret subsampling (const Image< I > &input, const typename I::site::delta`
`&first_p, const typename I::site::coord &gap)`
Subsampling FIXME : doxy.

9.137.1 Detailed Description

Namespace of "point-wise" expression tools.

9.137.2 Function Documentation

- 9.137.2.1** `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::antialiased`
`(const Image< I > & input, unsigned factor, const typename I::domain_t &`
`output_domain, unsigned border_thickness) [inline]`

Antialiased subsampling.

Parameters

- [in] *input* A gray-level image.
- [in] *factor* Subsampling ratio. Must be divisible by 2 or 3.
- [in] *output_domain* Force output domain.
- [in] *border_thickness* Force output border thickness.

Definition at line 418 of file antialiased.hh.

Referenced by antialiased().

- 9.137.2.2** `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::antialiased (`
`const Image< I > & input, unsigned factor) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 443 of file antialiased.hh.

References antialiased().

- 9.137.2.3** `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::gaussian_`
`subsampling (const Image< I > & input, float sigma, const typename I::dpsite &`
`first_p, const typename I::site::coord & gap) [inline]`

Gaussian subsampling FIXME : doxy.

Definition at line 62 of file gaussian_subsampling.hh.

References mln::linear::gaussian(), mln::geom::ncols(), and mln::geom::nrows().

9.137.2.4 `template<typename I> mln::trait::concrete< I>::ret mln::subsampling::subsampling
(const Image< I> & input, const typename I::site::delta & first_p, const typename
I::site::coord & gap) [inline]`

Subsampling FIXME : doxy.

Definition at line 91 of file subsampling.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.138 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

9.138.1 Detailed Description

Namespace of image processing routines related to tags.

9.139 mln::test Namespace Reference

Namespace of image processing routines related to pixel tests.

Namespaces

- namespace [impl](#)
Implementation namespace of test namespace.

Functions

- `template<typename I>`
`bool positive (const Image< I> &input)`
Test if an image only contains positive values.
- `template<typename S, typename F>`
`bool predicate (const Site_Set< S> &pset, const Function_v2b< F> &f)`
*Test if all points of *pset* verify the predicate *f*.*
- `template<typename I, typename J, typename F>`
`bool predicate (const Image< I> &lhs, const Image< J> &rhs, const Function_vv2b< F> &f)`
*Test if all pixel values of *lhs* and *rhs* verify the predicate *f*.*
- `template<typename I, typename F>`
`bool predicate (const Image< I> &ima, const Function_v2b< F> &f)`
*Test if all pixel values of *ima* verify the predicate *f*.*

9.139.1 Detailed Description

Namespace of image processing routines related to pixel tests.

9.139.2 Function Documentation

9.139.2.1 `template<typename I> bool mln::test::positive (const Image< I> & input)` [inline]

Test if an image only contains positive values.

Definition at line 54 of file `positive.hh`.

References `predicate()`, and `mln::literal::zero`.

Referenced by `mln::morpho::gradient()`, `mln::morpho::gradient_external()`, `mln::morpho::gradient_internal()`, `mln::morpho::top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `mln::morpho::top_hat_self_complementary()`, `mln::morpho::elementary::top_hat_self_complementary()`, `mln::morpho::top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

9.139.2.2 `template<typename S, typename F> bool mln::test::predicate (const Site_Set< S> & pset, const Function_v2b< F> & f)` [inline]

Test if all points of `pset` verify the predicate `f`.

Parameters

[in] *pset* The point set.

[in] *f* The predicate.

Definition at line 242 of file `predicate.hh`.

9.139.2.3 `template<typename I, typename J, typename F> bool mln::test::predicate (const Image< I> & lhs, const Image< J> & rhs, const Function_vv2b< F> & f)` [inline]

Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.

Parameters

[in] *lhs* The image.

[in] *rhs* The image.

[in] *f* The predicate.

Definition at line 222 of file `predicate.hh`.

9.139.2.4 `template<typename I, typename F> bool mln::test::predicate (const Image< I> & ima, const Function_v2b< F> & f)` [inline]

Test if all pixel values of `ima` verify the predicate `f`.

Parameters

[in] *ima* The image.

[in] *f* The predicate.

Definition at line 207 of file predicate.hh.

Referenced by mln::operator<(), mln::operator<=(), mln::operator==(), and positive().

9.140 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

9.140.1 Detailed Description

Implementation namespace of test namespace.

9.141 mln::topo Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [adj_higher_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.
- class [adj_higher_dim_connected_n_face_fwd_iter](#)
Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.
- class [adj_higher_face_bkd_iter](#)
Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.
- class [adj_higher_face_fwd_iter](#)
Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.
- class [adj_lower_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.
- class [adj_lower_dim_connected_n_face_fwd_iter](#)
Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.
- class [adj_lower_face_bkd_iter](#)
Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

- class [adj_lower_face_fwd_iter](#)
Forward iterator on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.
- class [adj_lower_higher_face_bkd_iter](#)
Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.
- class [adj_lower_higher_face_fwd_iter](#)
Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.
- class [adj_m_face_bkd_iter](#)
Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.
- class [adj_m_face_fwd_iter](#)
Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.
- class [algebraic_face](#)
Algebraic face handle in a complex; the face dimension is dynamic.
- class [algebraic_n_face](#)
Algebraic N -face handle in a complex.
- class [center_only_iter](#)
Iterator on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.
- class [centered_bkd_iter_adapter](#)
Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.
- class [centered_fwd_iter_adapter](#)
Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.
- class [complex](#)
General complex of dimension D .
- class [face](#)
Face handle in a complex; the face dimension is dynamic.
- class [face_bkd_iter](#)
Backward iterator on all the faces of an `mln::complex<D>`.
- class [face_fwd_iter](#)
Forward iterator on all the faces of an `mln::complex<D>`.
- struct [is_n_face](#)
A functor testing wheter a `mln::complex_psite` is an N -face.
- struct [is_simple_2d_t](#)
Test if a point is simple or not.
- class [is_simple_cell](#)
A predicate for the simplicity of a point based on the collapse property of the attachment.

- class [n_face](#)
N-face handle in a complex.
- class [n_face_bkd_iter](#)
Backward iterator on all the faces of an `mln::complex<D>`.
- class [n_face_fwd_iter](#)
Forward iterator on all the faces of an `mln::complex<D>`.
- class [n_faces_set](#)
Set of face handles of dimension N.
- class [static_n_face_bkd_iter](#)
Backward iterator on all the N-faces of a `mln::complex<D>`.
- class [static_n_face_fwd_iter](#)
Forward iterator on all the N-faces of a `mln::complex<D>`.

Functions

- `template<unsigned D, typename G >`
`void detach (const complex_psite< D, G > &f, complex_image< D, G, bool > &ima)`
Detach the cell corresponding to f from ima.
- `template<unsigned D, typename G >`
`bool is_facet (const complex_psite< D, G > &f)`
Is f a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?
- `template<unsigned D>`
`algebraic_face< D > make_algebraic_face (const face< D > &f, bool sign)`
Create an algebraic face handle of a D-complex.
- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > make_algebraic_n_face (const n_face< N, D > &f, bool sign)`
Create an algebraic N-face handle of a D-complex.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_n_face< N, D > &f)`
Print an `mln::topo::algebraic_n_face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_face< D > &f)`
Print an `mln::topo::algebraic_face`.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const n_face< N, D > &f)`
Print an `mln::topo::n_face`.

- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const face< D > &f)`
Print an [mln::topo::face](#).

- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const complex< D > &c)`
Pretty print a complex.

- `template<unsigned D>`
`bool operator== (const complex< D > &lhs, const complex< D > &rhs)`
Compare two complexes for equality.

- `template<unsigned D>`
`algebraic_face< D > operator- (const face< D > &f)`
Inversion operators.

- `template<unsigned D>`
`bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Comparison of two instances of [mln::topo::algebraic_face](#).

- `template<unsigned D>`
`bool operator!= (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs different from rhs?

- `template<unsigned D>`
`bool operator< (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs “less” than rhs?

- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > operator- (const n_face< N, D > &f)`
Inversion operators.

- `template<unsigned N, unsigned D>`
`bool operator== (const algebraic_n_face< N, D > &lhs, const algebraic_n_face< N, D > &rhs)`
Comparison of two instances of [mln::topo::algebraic_n_face](#).

- `template<unsigned N, unsigned D>`
`bool operator!= (const algebraic_n_face< N, D > &lhs, const algebraic_n_face< N, D > &rhs)`
Is lhs different from rhs?

- `template<unsigned N, unsigned D>`
`bool operator< (const algebraic_n_face< N, D > &lhs, const algebraic_n_face< N, D > &rhs)`
Is lhs “less” than rhs?

- `template<unsigned D>`
`algebraic_n_face< 1, D > edge (const n_face< 0, D > &f1, const n_face< 0, D > &f2)`
Helpers.

- template<unsigned D>
bool **operator==** (const **face**< D > &lhs, const **face**< D > &rhs)
*Comparison of two instances of **mln::topo::face**.*
- template<unsigned D>
bool **operator!=** (const **face**< D > &lhs, const **face**< D > &rhs)
Is lhs different from rhs?
- template<unsigned D>
bool **operator<** (const **face**< D > &lhs, const **face**< D > &rhs)
Is lhs "less" than rhs?
- template<unsigned N, unsigned D>
bool **operator==** (const **n_face**< N, D > &lhs, const **n_face**< N, D > &rhs)
*Comparison of two instances of **mln::topo::n_face**.*
- template<unsigned N, unsigned D>
bool **operator!=** (const **n_face**< N, D > &lhs, const **n_face**< N, D > &rhs)
Is lhs different from rhs?
- template<unsigned N, unsigned D>
bool **operator<** (const **n_face**< N, D > &lhs, const **n_face**< N, D > &rhs)
Is lhs "less" than rhs?
- template<unsigned N, unsigned D>
n_faces_set< N, D > **operator+** (const **algebraic_n_face**< N, D > &f1, const **algebraic_n_face**< N, D > &f2)
Addition.
- template<unsigned N, unsigned D>
n_faces_set< N, D > **operator-** (const **algebraic_n_face**< N, D > &f1, const **algebraic_n_face**< N, D > &f2)
Subtraction.

9.141.1 Detailed Description

Namespace of "point-wise" expression tools.

9.141.2 Function Documentation

9.141.2.1 template<unsigned D, typename G > void **mln::topo::detach** (const **complex_psite**< D, G > & *f*, **complex_image**< D, G, bool > & *ima*) [inline]

Detach the cell corresponding to *f* from *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).
 ima is an image of Boolean values.

Definition at line 58 of file detach.hh.

References `mln::make::detachment()`, `mln::data::fill()`, and `is_facet()`.

9.141.2.2 `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge (const n_face< 0, D > & f1, const n_face< 0, D > & f2)`

Helpers.

Return the algebraic 1-face (edge) linking the 0-faces (vertices) $f1$ and $f2$. If there is no 1-face between $f1$ and $f2$, return an invalid 1-face.

Precondition

$f1$ and $f2$ must belong to the same complex.

Note: this routine assumes the complex is not degenerated, i.e.,

- it does not check that $f1$ and $f2$ are the only 0-faces adjacent to an hypothetical 1-face; it just checks that $f1$ and $f2$ share a common 1-face;
- if there are several adjacent 1-faces shared by $f1$ and $f2$ (if the complex is ill-formed), there is no guarantee on the returned 1-face (the current implementation return the first 1-face found, but client code should not rely on this implementation-defined behavior).

Definition at line 286 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::higher_dim_adj_faces()`.

9.141.2.3 `template<unsigned D, typename G > bool mln::topo::is_facet (const complex_psite< D, G > & f) [inline]`

Is f a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?

Definition at line 58 of file is_facet.hh.

Referenced by `mln::make::attachment()`, `mln::make::cell()`, `detach()`, and `mln::make::detachment()`.

9.141.2.4 `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face (const face< D > & f, bool sign)`

Create an algebraic face handle of a `D-complex`.

Definition at line 211 of file algebraic_face.hh.

9.141.2.5 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::make_algebraic_n_face (const n_face< N, D > & f, bool sign)`

Create an algebraic `N-face` handle of a `D-complex`.

Definition at line 213 of file algebraic_n_face.hh.

9.141.2.6 `template<unsigned D> bool mln::topo::operator!= (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file algebraic_face.hh.

References `mln::topo::face< D >::cplx()`.

9.141.2.7 `template<unsigned D> bool mln::topo::operator!= (const face< D > & lhs, const face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 394 of file face.hh.

References `mln::topo::face< D >::cplx()`.

9.141.2.8 `template<unsigned N, unsigned D> bool mln::topo::operator!= (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 291 of file n_face.hh.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.9 `template<unsigned N, unsigned D> bool mln::topo::operator!= (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.10 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator+ (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Addition.

Definition at line 206 of file `n_faces_set.hh`.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.11 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- (const n_face< N, D > & f)`

Inversion operators.

Definition at line 221 of file `algebraic_n_face.hh`.

9.141.2.12 `template<unsigned D> algebraic_face< D > mln::topo::operator- (const face< D > & f)`

Inversion operators.

Definition at line 219 of file `algebraic_face.hh`.

9.141.2.13 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator- (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Subtraction.

Definition at line 284 of file `n_faces_set.hh`.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.14 `template<unsigned N, unsigned D> bool mln::topo::operator< (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 301 of file `n_face.hh`.

9.141.2.15 `template<unsigned D> bool mln::topo::operator< (const face< D > & lhs, const face< D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 404 of file face.hh.

9.141.2.16 `template<unsigned D> bool mln::topo::operator< (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 260 of file algebraic_face.hh.

9.141.2.17 `template<unsigned N, unsigned D> bool mln::topo::operator< (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 261 of file algebraic_n_face.hh.

9.141.2.18 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const complex< D > & c) [inline]`

Pretty print a complex.

Definition at line 670 of file complex.hh.

References `mln::topo::complex< D >::print()`.

9.141.2.19 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const face< D > & f) [inline]`

Print an [mln::topo::face](#).

Definition at line 416 of file face.hh.

9.141.2.20 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const n_face< N, D > & f) [inline]`

Print an [mln::topo::n_face](#).

Definition at line 312 of file n_face.hh.

9.141.2.21 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_face< D > & f) [inline]`

Print an [mln::topo::algebraic_face](#).

Definition at line 273 of file algebraic_face.hh.

9.141.2.22 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_n_face< N, D > & f) [inline]`

Print an [mln::topo::algebraic_n_face](#).

Definition at line 273 of file algebraic_n_face.hh.

9.141.2.23 `template<unsigned D> bool mln::topo::operator== (const face< D > & lhs, const face< D > & rhs) [inline]`

Comparison of two instances of [mln::topo::face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 384 of file face.hh.

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face_id\(\)](#), and [mln::topo::face< D >::n\(\)](#).

9.141.2.24 `template<unsigned N, unsigned D> bool mln::topo::operator== (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Comparison of two instances of [mln::topo::n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 281 of file n_face.hh.

References [mln::topo::n_face< N, D >::cplx\(\)](#), and [mln::topo::n_face< N, D >::face_id\(\)](#).

9.141.2.25 `template<unsigned N, unsigned D> bool mln::topo::operator== (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Comparison of two instances of [mln::topo::algebraic_n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 239 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::cplx()`, `mln::topo::n_face< N, D >::face_id()`, and `mln::topo::algebraic_n_face< N, D >::sign()`.

9.141.2.26 `template<unsigned D> bool mln::topo::operator==(const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Comparison of two instances of `mln::topo::algebraic_face`.

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 237 of file algebraic_face.hh.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, `mln::topo::face< D >::n()`, and `mln::topo::algebraic_face< D >::sign()`.

9.141.2.27 `template<unsigned D> bool mln::topo::operator==(const complex< D > & lhs, const complex< D > & rhs) [inline]`

Compare two complexes for equality.

Definition at line 657 of file complex.hh.

9.142 mln::trace Namespace Reference

Namespace of routines related to the trace mechanism.

9.142.1 Detailed Description

Namespace of routines related to the trace mechanism.

9.143 mln::trait Namespace Reference

Namespace where traits are defined.

9.143.1 Detailed Description

Namespace where traits are defined. Namespace for image traits.

9.144 mln::transform Namespace Reference

Namespace of transforms.

Functions

- `template<typename I , typename N , typename D >`
`util::couple< mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`
`distance_and_closest_point_geodesic (const Image< I > &input, const Neighborhood< N > &nbh,`
`D max)`
Discrete geodesic distance transform.
- `template<typename P , typename N , typename D >`
`util::couple< mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),`
`unsigned)> distance_and_closest_point_geodesic (const p_array< P > &pset, const box< P`
`> &closest_point_domain, const Neighborhood< N > &nbh, D max)`
Discrete geodesic distance transform.
- `template<typename I , typename N , typename D >`
`util::couple< mln::trait::ch_value< I, D >::ret, I > distance_and_influence_zone_geodesic (const`
`Image< I > &input, const Neighborhood< N > &nbh, D max)`
Discrete geodesic distance transform.
- `template<typename I , typename N , typename W , typename D >`
`mln::trait::ch_value< I, D >::ret distance_front (const Image< I > &input, const Neighborhood<`
`N > &nbh, const Weighted_Window< W > &w_win, D max)`
Discrete front distance transform.
- `template<typename I , typename N , typename D >`
`mln::trait::ch_value< I, D >::ret distance_geodesic (const Image< I > &input, const Neighbor-`
`hood< N > &nbh, D max)`
Discrete geodesic distance transform.
- `template<typename I >`
`image2d< float > hough (const Image< I > &input_)`
Compute the hough transform from a binary image.
- `template<typename I , typename N , typename W >`
`mln::trait::concrete< I >::ret influence_zone_front (const Image< I > &input, const Neighbor-`
`hood< N > &nbh, const Weighted_Window< W > &w_win)`
Influence zone transform.
- `template<typename I , typename N , typename W , typename D >`
`mln::trait::concrete< I >::ret influence_zone_front (const Image< I > &input, const Neighbor-`
`hood< N > &nbh, const Weighted_Window< W > &w_win, D max)`
Influence zone transform.
- `template<typename I , typename N >`
`mln::trait::concrete< I >::ret influence_zone_geodesic (const Image< I > &input, const Neighbor-`
`hood< N > &nbh)`
Geodesic influence zone transform.
- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const`
`Neighborhood< N > &nbh, const D &max, const typename I::value &background_value)`
Geodesic influence zone transform.

- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const`
`Neighborhood< N > &nbh, const D &max)`

9.144.1 Detailed Description

Namespace of transforms.

9.144.2 Function Documentation

9.144.2.1 `template<typename I , typename N , typename D > util::couple<`
`mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`
`mln::transform::distance_and_closest_point_geodesic (const Image< I > & input,`
`const Neighborhood< N > & nbh, D max) \[inline\]`

Discrete geodesic distance transform.

Parameters

- [in] *input* [Image](#) from which the geodesic distance is computed.
- [in] *nbh* [Neighborhood](#)
- [in] *max* Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as *input*.

Definition at line 90 of file `distance_and_closest_point_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

9.144.2.2 `template<typename P , typename N , typename D > util::couple<`
`mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),`
`unsigned)> mln::transform::distance_and_closest_point_geodesic (const p_array< P`
`> & pset, const box< P > & closest_point_domain, const Neighborhood< N > & nbh,`
`D max) \[inline\]`

Discrete geodesic distance transform.

Parameters

- [in] *pset* an array of sites.
- [in] *closest_point_domain* domain of the returned image.
- [in] *nbh* neighborhood
- [in] *max* max distance of propagation.

Returns

A couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains site indexes.

Postcondition

The returned image domains are defined on `closest_point_domain`.

Definition at line 110 of file `distance_and_closest_point_geodesic.hh`.

References `mln::geom::bbox()`, `mln::make::couple()`, `distance_geodesic()`, `mln::data::fill()`, and `mln::box<P>::is_valid()`.

9.144.2.3 `template<typename I, typename N, typename D > util::couple< mln::trait::ch_value< I, D >::ret, I > mln::transform::distance_and_influence_zone_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max) [inline]`

Discrete geodesic distance transform.

Parameters

[in] *input* Image from which the geodesic distance is computed.

[in] *nbh* Neighborhood

[in] *max* Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as `input`.

Definition at line 69 of file `distance_and_influence_zone_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

9.144.2.4 `template<typename I, typename N, typename W, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max) [inline]`

Discrete front distance transform.

Definition at line 56 of file `transform/distance_front.hh`.

9.144.2.5 `template<typename I, typename N, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max) [inline]`

Discrete geodesic distance transform.

Definition at line 55 of file `transform/distance_geodesic.hh`.

Referenced by `distance_and_closest_point_geodesic()`, and `distance_and_influence_zone_geodesic()`.

9.144.2.6 `template<typename I> image2d< float> mln::transform::hough (const Image< I> & input_)`

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

Parameters

[in] *input_* A binary image.

Returns

A 2D image of float. Rows are used for the distance and columns are used for the angles. Angles go from 0 to 359. Distance goes from 0 to the maximum distance between the center and a corner. The site having the maximum value indicates through its column index the document inclination.

Definition at line 98 of file `hough.hh`.

References `mln::opt::at()`, `mln::data::fill()`, `mln::geom::min_col()`, `mln::geom::min_row()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.144.2.7 `template<typename I, typename N, typename W> mln::trait::concrete< I>::ret mln::transform::influence_zone_front (const Image< I> & input, const Neighborhood< N> & nbh, const Weighted_Window< W> & w_win)`

Influence zone transform.

Definition at line 78 of file `influence_zone_front.hh`.

References `influence_zone_front()`.

9.144.2.8 `template<typename I, typename N, typename W, typename D> mln::trait::concrete< I>::ret mln::transform::influence_zone_front (const Image< I> & input, const Neighborhood< N> & nbh, const Weighted_Window< W> & w_win, D max)`

Influence zone transform.

Definition at line 60 of file `influence_zone_front.hh`.

References `mln::canvas::distance_front()`.

Referenced by `influence_zone_front()`.

9.144.2.9 `template<typename I, typename N> mln::trait::concrete< I>::ret mln::transform::influence_zone_geodesic (const Image< I> & input, const Neighborhood< N> & nbh)`

Geodesic influence zone transform.

Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

Returns

An image of influence zone.

Definition at line 216 of file `influence_zone_geodesic.hh`.

9.144.2.10 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret
mln::transform::influence_zone_geodesic_saturated (const Image< I > & input,
const Neighborhood< N > & nbh, const D & max, const typename I::value &
background_value)`

Geodesic influence zone transform.

Parameters

- [in] *input* An image.
- [in] *nbh* A neighborhood.
- [in] *max* The maximum influence zone distance.
- [in] *background_value* The value used as background (i.e. not propagated).

Returns

An image of influence zone.

Definition at line 73 of file `influence_zone_geodesic_saturated.hh`.

References `mln::canvas::distance_geodesic()`.

Referenced by `influence_zone_geodesic_saturated()`.

9.144.2.11 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret
mln::transform::influence_zone_geodesic_saturated (const Image< I > & input,
const Neighborhood< N > & nbh, const D & max)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 93 of file `influence_zone_geodesic_saturated.hh`.

References `influence_zone_geodesic_saturated()`, and `mln::literal::zero`.

9.145 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

Namespaces

- namespace [impl](#)
Implementation namespace of util namespace.

Classes

- class [adjacency_matrix](#)
A class of adjacency matrix.

- class [array](#)
A dynamic array class.
- class [branch](#)
Class of generic branch.
- class [branch_iter](#)
Basic 2D image class.
- class [branch_iter_ind](#)
Basic 2D image class.
- class [couple](#)
Definition of a couple.
- struct [eat](#)
Eat structure.
- class [edge](#)
Edge of a graph G .
- class [fibonacci_heap](#)
Fibonacci heap.
- class [graph](#)
Undirected graph.
- class [greater_point](#)
A “greater than” functor comparing points w.r.t.
- class [greater_psite](#)
A “greater than” functor comparing psites w.r.t.
- class [head](#)
Top structure of the soft heap.
- struct [ignore](#)
Ignore structure.
- struct [ilcell](#)
Element of an item list. Store the data (key) used in [soft_heap](#).
- class [line_graph](#)
Undirected line graph of a graph of type G .
- struct [nil](#)
Nil structure.
- class [node](#)

Meta-data of an element in the heap.

- class [object_id](#)

Base class of an object id.

- struct [ord](#)

Function-object that defines an ordering between objects with type T : lhs R rhs.

- struct [ord_pair](#)

Ordered pair structure s.a.

- struct [pix](#)

Structure pix.

- class [set](#)

An "efficient" mathematical set class.

- class [site_pair](#)

A pair of sites.

- class [soft_heap](#)

Soft heap.

- class [timer](#)

Timer structure.

- struct [tracked_ptr](#)

Smart pointer for shared data with tracking.

- class [tree](#)

Class of generic tree.

- class [tree_node](#)

Class of generic [tree_node](#) for tree.

- class [vertex](#)

[Vertex](#) of a graph G .

- struct [yes](#)

[Object](#) that always says "yes".

Typedefs

- typedef [object_id](#)< [vertex_tag](#), unsigned > [vertex_id_t](#)

[Vertex](#) id type.

Functions

- template<typename I , typename J >
void [display_branch](#) (const [Image](#)< J > &ima_, [tree_node](#)< I > *tree_node)
Display an arborescence from tree_node.
- template<typename I , typename J >
void [display_tree](#) (const [Image](#)< J > &ima_, [tree](#)< I > &tree)
Display a tree.
- template<typename I >
I::psite [lemmings](#) (const [Image](#)< I > &ima, const typename I::psite &pt, const typename I::psite::delta &dpt, const typename I::value &val)
Launch a lemmings on an image.
- template<typename I >
[greater_point](#)< I > [make_greater_point](#) (const [Image](#)< I > &ima)
Helper to build a mln::util::greater_point.
- template<typename I >
[greater_psite](#)< I > [make_greater_psite](#) (const [Image](#)< I > &ima)
Helper to build a mln::util::greater_psite.
- template<typename G >
bool [operator<](#) (const [vertex](#)< G > &lhs, const [vertex](#)< G > &rhs)
Less operator. Test whether lhs.id() < rhs.id().
- template<typename G >
std::ostream & [operator<<](#) (std::ostream &ostr, const [vertex](#)< G > &v)
Push the vertex v in the output stream ostr.
- template<typename T >
std::ostream & [operator<<](#) (std::ostream &ostr, const [array](#)< T > &a)
Operator<<.
- template<typename G >
bool [operator==](#) (const [vertex](#)< G > &v1, const [vertex](#)< G > &v2)
Equality operator.
- template<typename T >
bool [operator==](#) (const [array](#)< T > &lhs, const [array](#)< T > &rhs)
Operator==.
- template<typename T >
bool [ord_strict](#) (const T &lhs, const T &rhs)
Routine to test if lhs is strictly "less-than" rhs.
- template<typename T >
bool [ord_weak](#) (const T &lhs, const T &rhs)
Routine to test if lhs is "less-than or equal-to" rhs.

- `template<typename T, typename I>`
`void tree_fast_to_image (tree_fast< T> &tree, Image< I> &output_)`
- `template<typename T>`
`tree_fast< T> tree_to_fast (tree< T> &input)`
Facade.
- `template<typename T, typename I>`
`void tree_to_image (tree< T> &tree, Image< I> &output_)`
Convert a tree into an image.

9.145.1 Detailed Description

Namespace of tools using for more complex algorithm. Forward declaration.

9.145.2 Typedef Documentation

9.145.2.1 `typedef object_id<vertex_tag, unsigned> mln::util::vertex_id_t`

[Vertex](#) id type.

Definition at line 43 of file `graph_ids.hh`.

9.145.3 Function Documentation

9.145.3.1 `template<typename I, typename J> void mln::util::display_branch (const Image< J> & ima_, tree_node< I> * tree_node) [inline]`

Display an arborescence from [tree_node](#).

Parameters

- [in] *ima_* The domain of output image.
- [in] *tree_node* The root [tree_node](#) to display.

Definition at line 210 of file `tree_to_image.hh`.

References `mln::data::fill()`.

9.145.3.2 `template<typename I, typename J> void mln::util::display_tree (const Image< J> & ima_, tree< I> & tree) [inline]`

Display a tree.

Parameters

- [in] *ima_* The domain of output image.
- [in] *tree* The tree to display.

Definition at line 192 of file `tree_to_image.hh`.

References `mln::util::tree< T>::root()`.

9.145.3.3 `template<typename I> I::psite mln::util::lemmings (const Image< I> & ima, const typename I::psite & pt, const typename I::psite::delta & dpt, const typename I::value & val)`

Launch a lemmings on an image.

A lemmings is the point *pt* that you put on an image *ima*. This point will move through the image using the delta-point *dpt* while consider his value on the given image.

Returns

The first point that is not in the domain *domain* or which value on the given image is different to the value *val*.

Precondition

The domain *domain* must be contained in the domain of *ima*.

Definition at line 104 of file *lemmings.hh*.

9.145.3.4 `template<typename I> greater_point< I> mln::util::make_greater_point (const Image< I> & ima)`

Helper to build a [mln::util::greater_point](#).

Definition at line 82 of file *greater_point.hh*.

9.145.3.5 `template<typename I> greater_psite< I> mln::util::make_greater_psite (const Image< I> & ima)`

Helper to build a [mln::util::greater_psite](#).

Definition at line 82 of file *greater_psite.hh*.

9.145.3.6 `template<typename G> bool mln::util::operator< (const vertex< G> & lhs, const vertex< G> & rhs) [inline]`

Less operator. Test whether *lhs.id()* < *rhs.id()*.

Definition at line 390 of file *vertex.hh*.

9.145.3.7 `template<typename G> std::ostream & mln::util::operator<< (std::ostream & ostr, const vertex< G> & v) [inline]`

Push the vertex *v* in the output stream *ostr*.

Definition at line 372 of file *vertex.hh*.

9.145.3.8 `template<typename T> std::ostream & mln::util::operator<< (std::ostream & ostr, const array< T> & a)`

Operator<<.

Definition at line 796 of file *util/array.hh*.

References *mln::util::array< T>::nelements()*.

9.145.3.9 `template<typename G > bool mln::util::operator==(const vertex< G > & v1, const vertex< G > & v2) [inline]`

Equality operator.

Test whether two vertices have the same id.

Definition at line 380 of file vertex.hh.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

9.145.3.10 `template<typename T > bool mln::util::operator==(const array< T > & lhs, const array< T > & rhs)`

Operator==.

Definition at line 815 of file util/array.hh.

References `mln::util::array< T >::std_vector()`.

9.145.3.11 `template<typename T > bool mln::util::ord_strict (const T & lhs, const T & rhs) [inline]`

Routine to test if *lhs* is strictly "less-than" *rhs*.

Definition at line 90 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, and `mln::util::ord_pair< T >::change_second()`.

9.145.3.12 `template<typename T > bool mln::util::ord_weak (const T & lhs, const T & rhs) [inline]`

Routine to test if *lhs* is "less-than or equal-to" *rhs*.

Definition at line 101 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, `mln::util::ord_pair< T >::change_second()`, and `mln::box< P >::is_valid()`.

9.145.3.13 `template<typename T, typename I > void mln::util::tree_fast_to_image (tree_fast< T > & tree, Image< I > & output_) [inline]`

Convert a `tree_fast` into an image.

Parameters

[in] *tree* The tree to convert.

[out] *output_* The image containing tree informations.

Definition at line 99 of file tree_fast_to_image.hh.

9.145.3.14 `template<typename T > tree_fast< T > mln::util::tree_to_fast (tree< T > & input) [inline]`

Facade.

Convert a tree into an tree_fast.

Parameters

[in] *input* The tree to convert.

Returns

The tree_fast containing tree informations.

Definition at line 90 of file tree_to_fast.hh.

References mln::util::tree< T >::root().

9.145.3.15 `template<typename T , typename I > void mln::util::tree_to_image (tree< T > & tree, Image< I > & output_) [inline]`

Convert a tree into an image.

Parameters

[in] *tree* The tree to convert.

[out] *output_* The image containing tree information.

Definition at line 178 of file tree_to_image.hh.

9.146 mln::util::impl Namespace Reference

Implementation namespace of util namespace.

9.146.1 Detailed Description

Implementation namespace of util namespace.

9.147 mln::value Namespace Reference

Namespace of materials related to pixel value types.

Namespaces

- namespace [impl](#)
Implementation namespace of value namespace.

Classes

- class [float01](#)
Class for floating values restricted to the interval [0..1] and discretized with n bits.

- struct [float01_f](#)
Class for floating values restricted to the interval [0..1].
- struct [graylevel](#)
General gray-level class on n bits.
- struct [graylevel_f](#)
General gray-level class on n bits.
- struct [int_s](#)
Signed integer value class.
- struct [int_u](#)
Unsigned integer value class.
- struct [int_u_sat](#)
Unsigned integer value class with saturation behavior.
- struct [Integer](#)
Concept of integer.
- struct [Integer< void >](#)
Category flag type.
- struct [label](#)
Label value class.
- struct [lut_vec](#)
Class that defines FIXME.
- class [proxy](#)
Generic proxy class for an image pixel value.
- struct [rgb](#)
Color class for red-green-blue where every component is n-bit encoded.
- struct [set](#)
Class that defines the set of values of type T.
- class [sign](#)
The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.
- struct [stack_image](#)
Stack image class.
- struct [super_value< sign >](#)
Specializations:

- struct [value_array](#)
Generic array class over indexed by a value set with type T.

Typedefs

- typedef [float01_< 16 > float01_16](#)
Alias for 16 bit [float01](#).
- typedef [float01_< 8 > float01_8](#)
Alias for 8 bit [float01](#).
- typedef [graylevel< 16 > gl16](#)
Alias for 16 bit graylevel.
- typedef [graylevel< 8 > gl8](#)
Alias for 8 bit graylevel.
- typedef [graylevel_f glf](#)
Alias for graylevels encoded by float.
- typedef [int_s< 16 > int_s16](#)
Alias for signed 16-bit integers.
- typedef [int_s< 32 > int_s32](#)
Alias for signed 32-bit integers.
- typedef [int_s< 8 > int_s8](#)
Alias for signed 8-bit integers.
- typedef [int_u< 12 > int_u12](#)
Alias for unsigned 12-bit integers.
- typedef [int_u< 16 > int_u16](#)
Alias for unsigned 16-bit integers.
- typedef [mln::value::int_u< 32 > int_u32](#)
Alias for unsigned 32-bit integers.
- typedef [mln::value::int_u< 8 > int_u8](#)
Alias for unsigned 8-bit integers.
- typedef [label< 16 > label_16](#)
Alias for 16-bit integers.
- typedef [label< 32 > label_32](#)
Alias for 32-bit integers.
- typedef [mln::value::label< 8 > label_8](#)

Alias for 8-bit labels.

- typedef `rgb< 16 > rgb16`
Color class for red-green-blue where every component is 16-bit encoded.
- typedef `rgb< 8 > rgb8`
Color class for red-green-blue where every component is 8-bit encoded.

Functions

- template<typename Dest , typename Src >
Dest `cast` (const Src &src)
Cast a value `src` from type `Src` to type `Dest`.
- template<typename V >
internal::equiv_< V >::ret `equiv` (const mln::Value< V > &v)
Access to the equivalent value.
- template<unsigned n>
`rgb< n >::interop operator+` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)
Addition.
- template<typename H , typename S , typename L >
`hsl_< H, S, L > operator+` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)
Addition.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `label< n >` &l)
Print a label `l` into the output stream `ostr`.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `rgb< n >` &c)
Print an rgb `c` into the output stream `ostr`.
- std::ostream & `operator<<` (std::ostream &ostr, const `graylevel_f` &g)
Op<<.
- template<typename T >
std::ostream & `operator<<` (std::ostream &ostr, const scalar_< T > &s)
Print a scalar `s` in an output stream `ostr`.
- template<typename H , typename S , typename L >
std::ostream & `operator<<` (std::ostream &ostr, const `hsl_< H, S, L >` &c)
Print an hsl `c` into the output stream `ostr`.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `graylevel< n >` &g)
Op<<.

- template<unsigned n>
std::ostream & **operator<<** (std::ostream &ostr, const float01_< n > &f)
Op<<.
- std::ostream & **operator<<** (std::ostream &ostr, const **sign** &i)
Print an signed integer i into the output stream ostr.
- template<unsigned n>
std::ostream & **operator<<** (std::ostream &ostr, const **int_u**< n > &i)
Print an unsigned integer i into the output stream ostr.
- template<unsigned n>
std::ostream & **operator<<** (std::ostream &ostr, const **int_s**< n > &i)
Print an signed integer i into the output stream ostr.
- template<unsigned n>
std::ostream & **operator<<** (std::ostream &ostr, const **int_u_sat**< n > &i)
Print a saturated unsigned integer i into the output stream ostr.
- bool **operator==** (const **sign** &lhs, const **sign** &rhs)
Comparison operator.
- template<typename V >
V **other** (const V &val)
Give an other value than val.
- template<typename H , typename S , typename L >
hsl_< H, S, L > **operator-** (const hsl_< H, S, L > &lhs, const hsl_< H, S, L > &rhs)
Subtraction.
- template<typename H , typename S , typename L , typename S2 >
hsl_< H, S, L > **operator*** (const hsl_< H, S, L > &lhs, const mln::value::scalar_< S2 > &s)
Product.
- template<typename H , typename S , typename L , typename S2 >
hsl_< H, S, L > **operator/** (const hsl_< H, S, L > &lhs, const mln::value::scalar_< S2 > &s)
Division.
- template<typename H , typename S , typename L >
bool **operator==** (const hsl_< H, S, L > &lhs, const hsl_< H, S, L > &rhs)
Comparison.
- template<unsigned n>
rgb< n >::interop **operator-** (const **rgb**< n > &lhs, const **rgb**< n > &rhs)
Subtraction.

- `template<unsigned n, typename S >`
`rgb< n >::interop operator*` (`const rgb< n > &lhs`, `const mln::value::scalar_< S > &s`)
Product.
- `template<unsigned n, typename S >`
`rgb< n >::interop operator/` (`const rgb< n > &lhs`, `const mln::value::scalar_< S > &s`)
Division.
- `template<typename I >`
`stack_image< 2, const I > stack` (`const Image< I > &ima1`, `const Image< I > &ima2`)
Shortcut to build a stack with two images.

9.147.1 Detailed Description

Namespace of materials related to pixel value types.

9.147.2 Typedef Documentation

9.147.2.1 `typedef float01_<16> mln::value::float01_16`

Alias for 16 bit `float01`.

Definition at line 45 of file `float01_16.hh`.

9.147.2.2 `typedef float01_<8> mln::value::float01_8`

Alias for 8 bit `float01`.

Definition at line 45 of file `float01_8.hh`.

9.147.2.3 `typedef graylevel<16> mln::value::gl16`

Alias for 16 bit `graylevel`.

Definition at line 45 of file `gl16.hh`.

9.147.2.4 `typedef graylevel<8> mln::value::gl8`

Alias for 8 bit `graylevel`.

Definition at line 45 of file `gl8.hh`.

9.147.2.5 `typedef graylevel_f mln::value::glf`

Alias for graylevels encoded by float.

Definition at line 44 of file `glf.hh`.

9.147.2.6 typedef int_s<16> mln::value::int_s16

Alias for signed 16-bit integers.

Definition at line 45 of file int_s16.hh.

9.147.2.7 typedef int_s<32> mln::value::int_s32

Alias for signed 32-bit integers.

Definition at line 45 of file int_s32.hh.

9.147.2.8 typedef int_s<8> mln::value::int_s8

Alias for signed 8-bit integers.

Definition at line 45 of file int_s8.hh.

9.147.2.9 typedef int_u<12> mln::value::int_u12

Alias for unsigned 12-bit integers.

Definition at line 45 of file int_u12.hh.

9.147.2.10 typedef int_u<16> mln::value::int_u16

Alias for unsigned 16-bit integers.

Definition at line 45 of file int_u16.hh.

9.147.2.11 typedef mln::value::int_u<32> mln::value::int_u32

Alias for unsigned 32-bit integers.

Definition at line 45 of file int_u32.hh.

9.147.2.12 typedef mln::value::int_u<8> mln::value::int_u8

Alias for unsigned 8-bit integers.

Definition at line 44 of file int_u8.hh.

9.147.2.13 typedef label<16> mln::value::label_16

Alias for 16-bit integers.

Definition at line 44 of file label_16.hh.

9.147.2.14 typedef label<32> mln::value::label_32

Alias for 32-bit integers.

Definition at line 44 of file label_32.hh.

9.147.2.15 typedef mln::value::label<8> mln::value::label_8

Alias for 8-bit labels.

Definition at line 44 of file label_8.hh.

9.147.2.16 typedef rgb<16> mln::value::rgb16

Color class for red-green-blue where every component is 16-bit encoded.

Definition at line 45 of file rgb16.hh.

9.147.2.17 typedef rgb<8> mln::value::rgb8

Color class for red-green-blue where every component is 8-bit encoded.

Definition at line 45 of file rgb8.hh.

9.147.3 Function Documentation**9.147.3.1 template<typename Dest , typename Src > Dest mln::value::cast (const Src & src) [inline]**

Cast a value `src` from type `Src` to type `Dest`.

Definition at line 85 of file value/cast.hh.

9.147.3.2 template<typename V > internal::equiv_< V >::ret mln::value::equiv (const mln::Value< V > & v) [inline]

Access to the equivalent value.

Definition at line 153 of file equiv.hh.

Referenced by `mln::labeling::superpose()`.

9.147.3.3 template<unsigned n, typename S > rgb< n >::interop mln::value::operator* (const rgb< n > & lhs, const mln::value::scalar_< S > & s) [inline]

Product.

Definition at line 717 of file value/rgb.hh.

9.147.3.4 template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator* (const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s)

Product.

Definition at line 357 of file hsl.hh.

9.147.3.5 `template<unsigned n> rgb< n >::interop mln::value::operator+ (const rgb< n > & lhs, const rgb< n > & rhs) [inline]`

Addition.

{

Definition at line 663 of file value/rgb.hh.

9.147.3.6 `template<typename H , typename S , typename L > hsl< H, S, L > mln::value::operator+ (const hsl< H, S, L > & lhs, const hsl< H, S, L > & rhs)`

Addition.

{

Definition at line 337 of file hsl.hh.

9.147.3.7 `template<unsigned n> rgb< n >::interop mln::value::operator- (const rgb< n > & lhs, const rgb< n > & rhs) [inline]`

Subtraction.

Definition at line 690 of file value/rgb.hh.

9.147.3.8 `template<typename H , typename S , typename L > hsl< H, S, L > mln::value::operator- (const hsl< H, S, L > & lhs, const hsl< H, S, L > & rhs)`

Subtraction.

Definition at line 347 of file hsl.hh.

9.147.3.9 `template<unsigned n, typename S > rgb< n >::interop mln::value::operator/ (const rgb< n > & lhs, const mln::value::scalar_< S > & s) [inline]`

Division.

Definition at line 735 of file value/rgb.hh.

9.147.3.10 `template<typename H , typename S , typename L , typename S2 > hsl< H, S, L > mln::value::operator/ (const hsl< H, S, L > & lhs, const mln::value::scalar_< S2 > & s)`

Division.

Definition at line 367 of file hsl.hh.

9.147.3.11 `template<typename T > std::ostream & mln::value::operator<< (std::ostream & ostr, const scalar_< T > & s) [inline]`

Print a scalar *s* in an output stream *ostr*.

Definition at line 130 of file scalar.hh.

9.147.3.12 `std::ostream & mln::value::operator<< (std::ostream & ostr, const sign & i)`
`[inline]`

Print an signed integer `i` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An sign value

Returns

The modified output stream `ostr`.

Definition at line 184 of file `value/sign.hh`.

References `mln::debug::format()`.

9.147.3.13 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr,`
`const int_s<n> & i) [inline]`

Print an signed integer `i` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An signed integer.

Returns

The modified output stream `ostr`.

Definition at line 260 of file `int_s.hh`.

References `mln::debug::format()`.

9.147.3.14 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr,`
`const graylevel<n> & g) [inline]`

`Op<<.`

Definition at line 591 of file `graylevel.hh`.

9.147.3.15 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr,`
`const int_u<n> & i) [inline]`

Print an unsigned integer `i` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An unsigned integer.

Returns

The modified output stream `ostr`.

Definition at line 357 of file `int_u.hh`.

References `mln::debug::format()`.

9.147.3.16 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const int_u_sat< n > & i) [inline]`

Print a saturated unsigned integer `i` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *i* A saturated unsigned integer.

Returns

The modified output stream `ostr`.

Definition at line 220 of file `int_u_sat.hh`.

References `mln::debug::format()`.

9.147.3.17 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const rgb< n > & c) [inline]`

Print an rgb `c` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

Returns

The modified output stream `ostr`.

Definition at line 743 of file `value/rgb.hh`.

References `mln::debug::format()`.

9.147.3.18 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const float01_< n > & f) [inline]`

Op<<.

Definition at line 253 of file `float01_.hh`.

9.147.3.19 `template<typename H , typename S , typename L > std::ostream & mln::value::operator<< (std::ostream & ostr, const hsl_< H, S, L > & c) [inline]`

Print an hsl *c* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

Returns

The modified output stream *ostr*.

Definition at line 326 of file hsl.hh.

References `mln::debug::format()`.

9.147.3.20 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const label< n > & l) [inline]`

Print a label *l* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *l* A label.

Returns

The modified output stream *ostr*.

Definition at line 353 of file label.hh.

References `mln::debug::format()`.

9.147.3.21 `std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel_f & g) [inline]`

Op<<.

Definition at line 458 of file graylevel_f.hh.

References `mln::value::graylevel_f::value()`.

9.147.3.22 `template<typename H , typename S , typename L > bool mln::value::operator==(const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs)`

Comparison.

Definition at line 376 of file hsl.hh.

9.147.3.23 `bool mln::value::operator==(const sign & lhs, const sign & rhs) [inline]`

Comparison operator.

Definition at line 190 of file value/sign.hh.

9.147.3.24 `template<typename V > V mln::value::other (const V & val) [inline]`

Give an other value than `val`.

Definition at line 115 of file other.hh.

9.147.3.25 `template<typename I > stack_image< 2, const I > mln::value::stack (const Image< I > & ima1, const Image< I > & ima2) [inline]`

Shortcut to build a stack with two images.

Definition at line 306 of file stack.hh.

9.148 mln::value::impl Namespace Reference

Implementation namespace of value namespace.

9.148.1 Detailed Description

Implementation namespace of value namespace.

9.149 mln::win Namespace Reference

Namespace of image processing routines related to win.

Classes

- struct [backdiag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [ball](#)
Generic ball window defined on a given grid.
- struct [cube3d](#)
Cube window defined on the 3D grid.
- struct [cuboid3d](#)
Cuboid defined on the 3-D square grid.
- struct [diag2d](#)
Diagonal line window defined on the 2D square grid.

- struct [line](#)
Generic line window defined on a given grid in the given dimension.
- class [multiple](#)
Multiple window.
- class [multiple_size](#)
Definition of a multiple-size window.
- struct [octagon2d](#)
Octagon window defined on the 2D square grid.
- struct [rectangle2d](#)
Rectangular window defined on the 2D square grid.

Typedefs

- typedef [ball](#)< grid::square, [def::coord](#) > [disk2d](#)
2D disk window; precisely, ball-shaped window defined on the 2D square grid.
- typedef [line](#)< grid::square, 1, [def::coord](#) > [hline2d](#)
Horizontal line window defined on the 2D square grid.
- typedef [line](#)< grid::tick, 0, [def::coord](#) > [segment1d](#)
Segment window defined on the 1D grid.
- typedef [line](#)< grid::cube, 0, [def::coord](#) > [sline3d](#)
Depth line window defined on the 3D cubic grid.
- typedef [ball](#)< grid::cube, [def::coord](#) > [sphere3d](#)
3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.
- typedef [line](#)< grid::square, 0, [def::coord](#) > [vline2d](#)
Vertical line window defined on the 2D square grid.

Functions

- template<typename N1 , typename N2 >
[neighb](#)< typename N1::window::regular > [diff](#) (const [Neighborhood](#)< N1 > &nbh1, const [Neighborhood](#)< N2 > &nbh2)
Set difference between a couple of neighborhoods nbh1 and nbh2.
- template<typename W1 , typename W2 >
[mln_regular](#) (W1) [diff](#)(const [Window](#)< W1 > &win1
Set difference between a couple of windows win1 and win2.
- template<typename W >
[mln_regular](#) (W) [shift](#)(const [Window](#)< W > &win

Shift a window `win` with a delta-point `dp`.

- `template<typename W >`
`W sym (const Window< W > &win)`
Give the symmetrical window of `win`.
- `template<typename W >`
`W sym (const Weighted_Window< W > &w_win)`
Give the symmetrical weighted window of `w_win`.

9.149.1 Detailed Description

Namespace of image processing routines related to win.

9.149.2 Function Documentation

9.149.2.1 `template<typename N1 , typename N2 > N2 neighb< typename N1::window::regular >`
`mln::win::diff (const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > &`
`nbh2)`

Set difference between a couple of neighborhoods `nbh1` and `nbh2`.

Definition at line 132 of file win/diff.hh.

Referenced by `mln::operator()`.

9.149.2.2 `template<typename W1 , typename W2 > mln::win::mln_regular (W1) const`
`[inline]`

Set difference between a couple of windows `win1` and `win2`.

9.149.2.3 `template<typename W > mln::win::mln_regular (W) const [inline]`

Shift a window `win` with a delta-point `dp`.

9.149.2.4 `template<typename W > W mln::win::sym (const Window< W > & win)`
`[inline]`

Give the symmetrical window of `win`.

Definition at line 59 of file sym.hh.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c4_3d()`, `mln::c6()`, `mln::morpho::hit_or_miss_background_opening()`, `mln::morpho::hit_or_miss_opening()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.149.2.5 `template<typename W > W mln::win::sym (const Weighted_Window< W > & w_win`
`) [inline]`

Give the symmetrical weighted window of `w_win`.

Definition at line 71 of file sym.hh.

Chapter 10

Class Documentation

10.1 mln::accu::center< P, V > Struct Template Reference

Mass center accumulator.

```
#include <center.hh>
```

Inherits base< V, center< P, V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- unsigned [nsites](#) () const
Return the number of sites taken in consideration.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- V [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.1.1 Detailed Description

```
template<typename P, typename V = typename P::vec> struct mln::accu::center< P, V >
```

Mass center accumulator.

Template Parameters

P the type of site.

V the type of vector to be used as result. The default vector type is the one provided by P.

Definition at line 55 of file center.hh.

10.1.2 Member Function Documentation

10.1.2.1 `template<typename P , typename V > void mln::accu::center< P, V >::init ()`
[inline]

Manipulators.

Definition at line 116 of file center.hh.

References mln::literal::zero.

10.1.2.2 `template<typename P , typename V > bool mln::accu::center< P, V >::is_valid ()`
const [inline]

Check whether this accu is able to return a result.

Definition at line 160 of file center.hh.

Referenced by mln::accu::center< P, V >::to_result().

10.1.2.3 `template<typename P , typename V > unsigned mln::accu::center< P, V >::nsites ()`
const [inline]

Return the number of sites taken in consideration.

Definition at line 168 of file center.hh.

10.1.2.4 `void mln::Accumulator< center< P, V > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.1.2.5 `void mln::Accumulator< center< P, V > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.1.2.6 `template<typename P , typename V > V mln::accu::center< P, V >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 142 of file center.hh.

References mln::accu::center< P, V >::is_valid().

10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference

Generic convolution accumulator class.

```
#include <convolve.hh>
```

Inherits base< R, convolve< T1, T2, R > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- R [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.2.1 Detailed Description

```
template<typename T1, typename T2, typename R = typename mln::trait::value_< typename
mln::trait::op::times< T1 , T2 >::ret >::sum> struct mln::accu::convolve< T1, T2, R >
```

Generic convolution accumulator class. Parameters T1 and T2 are the type of values to be convolved. Parameter R is the result type.

Definition at line 54 of file accu/convolve.hh.

10.2.2 Member Function Documentation

10.2.2.1 template<typename T1 , typename T2 , typename R > void mln::accu::convolve< T1, T2, R >::init () [inline]

Manipulators.

Definition at line 96 of file accu/convolve.hh.

References mln::literal::zero.

10.2.2.2 `template<typename T1 , typename T2 , typename R > bool mln::accu::convolve< T1, T2, R >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 137 of file accu/convolve.hh.

10.2.2.3 `void mln::Accumulator< convolve< T1, T2, R > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.2.2.4 `void mln::Accumulator< convolve< T1, T2, R > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.2.2.5 `template<typename T1 , typename T2 , typename R > R mln::accu::convolve< T1, T2, R >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 129 of file accu/convolve.hh.

10.3 `mln::accu::count_adjacent_vertices< F, S > Struct Template Reference`

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count_adjacent_vertices< F, S > >.

Public Member Functions

- bool [is_valid](#) () const
Return whether this accu can return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- unsigned [to_result](#) () const

Get the value of the accumulator.

- void `init()`
Manipulators.
- void `set_value(unsigned c)`
Force the value of the counter to c.

10.3.1 Detailed Description

`template<typename F, typename S> struct mln::accu::count_adjacent_vertices< F, S >`

Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges). The type to be count is `mln::util::pix< pw::image<F, S> >` where `F` and `S` are the parameters of this class.

This accumulator is used by `mln::closing_area_on_vertices` and `mln::opening_area_on_vertices`.

Definition at line 58 of file `accu/count_adjacent_vertices.hh`.

10.3.2 Member Function Documentation

10.3.2.1 `template<typename F, typename S> void mln::accu::count_adjacent_vertices< F, S >::init() [inline]`

Manipulators.

Definition at line 123 of file `accu/count_adjacent_vertices.hh`.

10.3.2.2 `template<typename F, typename S> bool mln::accu::count_adjacent_vertices< F, S >::is_valid() const [inline]`

Return whether this accu can return a result.

Definition at line 177 of file `accu/count_adjacent_vertices.hh`.

10.3.2.3 `template<typename F, typename S> void mln::accu::count_adjacent_vertices< F, S >::set_value(unsigned c) [inline]`

Force the value of the counter to `c`.

Definition at line 159 of file `accu/count_adjacent_vertices.hh`.

10.3.2.4 `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init(const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.3.2.5 `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.3.2.6 `template<typename F, typename S> unsigned mln::accu::count_adjacent_vertices< F, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 151 of file `accu/count_adjacent_vertices.hh`.

10.4 mln::accu::count_labels< L > Struct Template Reference

Count the number of different labels in an image.

```
#include <count_labels.hh>
```

Inherits `base< unsigned, count_labels< L > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.
- `void set_value (unsigned c)`
Force the value of the counter to c .

10.4.1 Detailed Description

`template<typename L> struct mln::accu::count_labels< L >`

Count the number of different labels in an image. The parameter L is the label type to be count.

Definition at line 52 of file `count_labels.hh`.

10.4.2 Member Function Documentation

10.4.2.1 `template<typename L > void mln::accu::count_labels< L >::init () [inline]`

Manipulators.

Definition at line 113 of file count_labels.hh.

10.4.2.2 `template<typename L > bool mln::accu::count_labels< L >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 163 of file count_labels.hh.

10.4.2.3 `template<typename L > void mln::accu::count_labels< L >::set_value (unsigned c) [inline]`

Force the value of the counter to *c*.

Definition at line 155 of file count_labels.hh.

10.4.2.4 `void mln::Accumulator< count_labels< L > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.4.2.5 `void mln::Accumulator< count_labels< L > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.4.2.6 `template<typename L > unsigned mln::accu::count_labels< L >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 146 of file count_labels.hh.

10.5 mln::accu::count_value< V > Struct Template Reference

Define an accumulator that counts the occurrence of a given value.

```
#include <count_value.hh>
```

Inherits `base< unsigned, count_value< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.
- `void set_value (unsigned c)`
Force the value of the counter to c .

10.5.1 Detailed Description

`template<typename V> struct mln::accu::count_value< V >`

Define an accumulator that counts the occurrence of a given value.

Definition at line 72 of file count_value.hh.

10.5.2 Member Function Documentation

10.5.2.1 `template<typename V > void mln::accu::count_value< V >::init () [inline]`

Manipulators.

Definition at line 153 of file count_value.hh.

10.5.2.2 `template<typename V > bool mln::accu::count_value< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 216 of file count_value.hh.

10.5.2.3 `template<typename V > void mln::accu::count_value< V >::set_value (unsigned c) [inline]`

Force the value of the counter to c .

Definition at line 208 of file count_value.hh.

10.5.2.4 `void mln::Accumulator< count_value< V > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.5.2.5 `void mln::Accumulator< count_value< V > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.5.2.6 `template<typename V> unsigned mln::accu::count_value< V >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 199 of file `count_value.hh`.

10.6 mln::accu::histo< V > Struct Template Reference

Generic histogram class over a value set with type V .

`#include <histo.hh>`

Inherits `base< const std::vector< unsigned > &, histo< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `void take (const argument &t)`
Manipulators.
- `const std::vector< unsigned > & vect () const`
Get the value of the accumulator.

10.6.1 Detailed Description

template<typename V> struct mln::accu::histo< V >

Generic histogram class over a value set with type V.

Definition at line 56 of file accu/histo.hh.

10.6.2 Member Function Documentation

10.6.2.1 template<typename V > bool mln::accu::histo< V >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 236 of file accu/histo.hh.

10.6.2.2 template<typename V > void mln::accu::histo< V >::take (const argument & t) [inline]

Manipulators.

Definition at line 129 of file accu/histo.hh.

10.6.2.3 void mln::Accumulator< histo< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.6.2.4 void mln::Accumulator< histo< V > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.6.2.5 template<typename V > const std::vector< unsigned > & mln::accu::histo< V >::vect () const [inline]

Get the value of the accumulator.

Definition at line 201 of file accu/histo.hh.

10.7 mln::accu::label_used< L > Struct Template Reference

References all the labels used.

```
#include <label_used.hh>
```

Inherits base< const fun::i2v::array< bool > &, label_used< L > >.

Public Member Functions

- void [init](#) ()
Initialize accumulator attributes.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const fun::i2v::array< bool > & [to_result](#) () const
Get the value of the accumulator.
- void [take](#) (const argument &)
Manipulators.

10.7.1 Detailed Description

template<typename L> struct mln::accu::label_used< L >

References all the labels used. The parameter *L* is the label type.

Definition at line 53 of file label_used.hh.

10.7.2 Member Function Documentation

10.7.2.1 template<typename L > void mln::accu::label_used< L >::init () [inline]

Initialize accumulator attributes.

Definition at line 110 of file label_used.hh.

10.7.2.2 template<typename L > bool mln::accu::label_used< L >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 150 of file label_used.hh.

10.7.2.3 template<typename L > void mln::accu::label_used< L >::take (const argument & l) [inline]

Manipulators.

Definition at line 118 of file label_used.hh.

10.7.2.4 void mln::Accumulator< label_used< L > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.7.2.5 void mln::Accumulator< label_used< L > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.7.2.6 template<typename L > const fun::i2v::array< bool > & mln::accu::label_used< L >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 142 of file label_used.hh.

10.8 mln::accu::logic::land Struct Reference

"Logical-and" accumulator.

```
#include <land.hh>
```

Inherits base< bool, land >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- bool [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.8.1 Detailed Description

"Logical-and" accumulator.

Definition at line 96 of file accu/logic/land.hh.

10.8.2 Member Function Documentation

10.8.2.1 void mln::accu::logic::land::init () [inline]

Manipulators.

Definition at line 136 of file accu/logic/land.hh.

10.8.2.2 bool mln::accu::logic::land::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 185 of file accu/logic/land.hh.

10.8.2.3 void mln::Accumulator< land >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.8.2.4 void mln::Accumulator< land >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.8.2.5 bool mln::accu::logic::land::to_result () const [inline]

Get the value of the accumulator.

Definition at line 178 of file accu/logic/land.hh.

10.9 mln::accu::logic::land_basic Struct Reference

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits base< bool, land_basic >.

Public Member Functions

- bool [can_stop](#) () const

Test if it is worth for this accumulator to take extra data.

- bool `is_valid()` const

Check whether this accu is able to return a result.

- void `take_as_init` (const T &t)

Take as initialization the value t.

- void `take_n_times` (unsigned n, const T &t)

Take n times the value t.

- bool `to_result()` const

Get the value of the accumulator.

- void `init()`

Manipulators.

10.9.1 Detailed Description

"Logical-and" accumulator. Conversely to `accu::logic::land`, this version does not have the 'untake' method but features the 'can_stop' method.

Definition at line 99 of file `land_basic.hh`.

10.9.2 Member Function Documentation

10.9.2.1 bool `mln::accu::logic::land_basic::can_stop()` const [inline]

Test if it is worth for this accumulator to take extra data.

If the result is already 'false' (because this accumulator has already taken a 'false' value), `can_stop` returns true.

Definition at line 181 of file `land_basic.hh`.

10.9.2.2 void `mln::accu::logic::land_basic::init()` [inline]

Manipulators.

Definition at line 140 of file `land_basic.hh`.

10.9.2.3 bool `mln::accu::logic::land_basic::is_valid()` const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file `land_basic.hh`.

10.9.2.4 void mln::Accumulator< land_basic >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.9.2.5 void mln::Accumulator< land_basic >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.9.2.6 bool mln::accu::logic::land_basic::to_result () const [inline]

Get the value of the accumulator.

Definition at line 167 of file land_basic.hh.

10.10 mln::accu::logic::lor Struct Reference

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits base< bool, lor >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- bool [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.10.1 Detailed Description

"Logical-or" accumulator.

Definition at line 95 of file accu/logic/lor.hh.

10.10.2 Member Function Documentation

10.10.2.1 void mln::accu::logic::lor::init () [inline]

Manipulators.

Definition at line 134 of file accu/logic/lor.hh.

10.10.2.2 bool mln::accu::logic::lor::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 183 of file accu/logic/lor.hh.

10.10.2.3 void mln::Accumulator< lor >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.10.2.4 void mln::Accumulator< lor >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.10.2.5 bool mln::accu::logic::lor::to_result () const [inline]

Get the value of the accumulator.

Definition at line 176 of file accu/logic/lor.hh.

10.11 mln::accu::logic::lor_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits base< bool, lor_basic >.

Public Member Functions

- bool [can_stop](#) () const
Test if it is worth for this accumulator to take extra data.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)

Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t .

- bool [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

10.11.1 Detailed Description

"Logical-or" accumulator class. Conversely to [accu::logic::lor](#), this version does not have the 'untake' method but features the 'can_stop' method.

Definition at line 98 of file lor_basic.hh.

10.11.2 Member Function Documentation

10.11.2.1 bool mln::accu::logic::lor_basic::can_stop () const [inline]

Test if it is worth for this accumulator to take extra data.

If the result is already 'true' (because this accumulator has already taken a 'true' value), can_stop returns true.

Definition at line 180 of file lor_basic.hh.

10.11.2.2 void mln::accu::logic::lor_basic::init () [inline]

Manipulators.

Definition at line 139 of file lor_basic.hh.

10.11.2.3 bool mln::accu::logic::lor_basic::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 173 of file lor_basic.hh.

10.11.2.4 void mln::Accumulator<lor_basic>::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.11.2.5 void mln::Accumulator< lor_basic >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.11.2.6 bool mln::accu::logic::lor_basic::to_result () const [inline]

Get the value of the accumulator.

Definition at line 166 of file lor_basic.hh.

10.12 mln::accu::maj_h< T > Struct Template Reference

Compute the majority value.

```
#include <maj_h.hh>
```

Inherits base< const T &, maj_h< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.12.1 Detailed Description

```
template<typename T> struct mln::accu::maj_h< T >
```

Compute the majority value. It is based on a histogram. The parameter T is the type of values.

Definition at line 57 of file maj_h.hh.

10.12.2 Member Function Documentation

10.12.2.1 `template<typename T> void mln::accu::maj_h< T >::init () [inline]`

Manipulators.

Definition at line 129 of file maj_h.hh.

10.12.2.2 `template<typename T> bool mln::accu::maj_h< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file maj_h.hh.

10.12.2.3 `void mln::Accumulator< maj_h< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.12.2.4 `void mln::Accumulator< maj_h< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.12.2.5 `template<typename T> const T & mln::accu::maj_h< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 187 of file maj_h.hh.

10.13 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits `base< unsigned, count< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t.

- unsigned [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

- void [set_value](#) (unsigned c)

Force the value of the counter to c.

10.13.1 Detailed Description

template<typename T> struct mln::accu::math::count< T >

Generic counter accumulator. The parameter *T* is the type to be count.

Definition at line 100 of file count.hh.

10.13.2 Member Function Documentation

10.13.2.1 template<typename T > void mln::accu::math::count< T >::init () [inline]

Manipulators.

Definition at line 145 of file count.hh.

**10.13.2.2 template<typename T > bool mln::accu::math::count< T >::is_valid () const
 [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 203 of file count.hh.

**10.13.2.3 template<typename T > void mln::accu::math::count< T >::set_value (unsigned c)
 [inline]**

Force the value of the counter to *c*.

Definition at line 195 of file count.hh.

10.13.2.4 void mln::Accumulator< count< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.13.2.5 `void mln::Accumulator< count< T > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.13.2.6 `template<typename T > unsigned mln::accu::math::count< T >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 187 of file `count.hh`.

10.14 mln::accu::math::inf< T > Struct Template Reference

Generic inf accumulator class.

`#include <inf.hh>`

Inherits `base< const T &, inf< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
*Take as initialization the value *t*.*
- `void take_n_times (unsigned n, const T &t)`
*Take *n* times the value *t*.*
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.14.1 Detailed Description

`template<typename T> struct mln::accu::math::inf< T >`

Generic inf accumulator class. The parameter *T* is the type of values.

Definition at line 56 of file `accu/math/inf.hh`.

10.14.2 Member Function Documentation

10.14.2.1 `template<typename T> void mln::accu::math::inf< T>::init () [inline]`

Manipulators.

Definition at line 126 of file `accu/math/inf.hh`.

10.14.2.2 `template<typename T> bool mln::accu::math::inf< T>::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 164 of file `accu/math/inf.hh`.

10.14.2.3 `void mln::Accumulator< inf< T>>::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.14.2.4 `void mln::Accumulator< inf< T>>::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.14.2.5 `template<typename T> const T & mln::accu::math::inf< T>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 156 of file `accu/math/inf.hh`.

10.15 `mln::accu::math::sum< T, S>` Struct Template Reference

Generic sum accumulator class.

```
#include <sum.hh>
```

Inherits `base< const S &, sum< T, S>>`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.

- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const S & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.15.1 Detailed Description

template<typename T, typename S = typename mln::value::props< T >::sum> struct mln::accu::math::sum< T, S >

Generic sum accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the value sum; the default type of S is the summation type (property) of T.

Definition at line 112 of file accu/math/sum.hh.

10.15.2 Member Function Documentation

10.15.2.1 template<typename T, typename S > void mln::accu::math::sum< T, S >::init ()
[inline]

Manipulators.

Definition at line 161 of file accu/math/sum.hh.

References mln::literal::zero.

10.15.2.2 template<typename T, typename S > bool mln::accu::math::sum< T, S >::is_valid ()
const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 222 of file accu/math/sum.hh.

10.15.2.3 void mln::Accumulator< sum< T, S > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.15.2.4 void mln::Accumulator< sum< T, S > >::take_n_times (unsigned n, const T & t)
[inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.15.2.5 `template<typename T , typename S > const S & mln::accu::math::sum< T, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 206 of file `accu/math/sum.hh`.

10.16 mln::accu::math::sup< T > Struct Template Reference

Generic sup accumulator class.

```
#include <sup.hh>
```

Inherits `base< const T &, sup< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.16.1 Detailed Description

`template<typename T> struct mln::accu::math::sup< T >`

Generic sup accumulator class. The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/sup.hh`.

10.16.2 Member Function Documentation

10.16.2.1 `template<typename T > void mln::accu::math::sup< T >::init () [inline]`

Manipulators.

Definition at line 128 of file `accu/math/sup.hh`.

10.16.2.2 `template<typename T> bool mln::accu::math::sup< T >::is_valid () const`
[inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 166 of file accu/math/sup.hh.

10.16.2.3 `void mln::Accumulator< sup< T > >::take_as_init (const T & t)` **[inherited]**

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.16.2.4 `void mln::Accumulator< sup< T > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.16.2.5 `template<typename T> const T & mln::accu::math::sup< T >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 158 of file accu/math/sup.hh.

10.17 mln::accu::max_site< I > Struct Template Reference

Define an accumulator that computes the first site with the maximum value in an image.

```
#include <max_site.hh>
```

Inherits `base< I::psite, max_site< I > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `I::psite to_result () const`
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.17.1 Detailed Description

template<typename I> struct mln::accu::max_site< I >

Define an accumulator that computes the first site with the maximum value in an image.

Definition at line 53 of file max_site.hh.

10.17.2 Member Function Documentation

10.17.2.1 template<typename I > void mln::accu::max_site< I >::init () [inline]

Manipulators.

Definition at line 114 of file max_site.hh.

10.17.2.2 template<typename I > bool mln::accu::max_site< I >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file max_site.hh.

10.17.2.3 void mln::Accumulator< max_site< I > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.17.2.4 void mln::Accumulator< max_site< I > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.17.2.5 template<typename I > I::psite mln::accu::max_site< I >::to_result () const [inline]

Get the value of the accumulator.

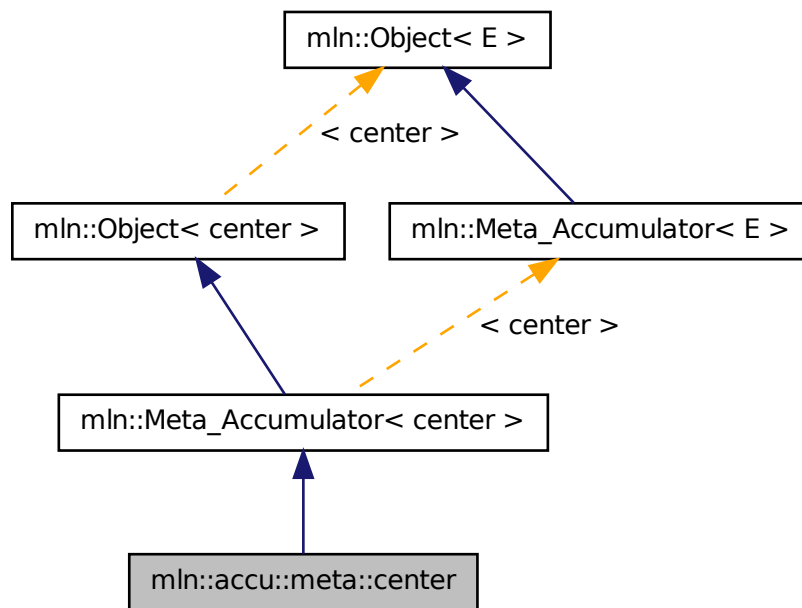
Definition at line 150 of file max_site.hh.

10.18 mln::accu::meta::center Struct Reference

Meta accumulator for center.

```
#include <center.hh>
```

Inheritance diagram for mln::accu::meta::center:



10.18.1 Detailed Description

Meta accumulator for center.

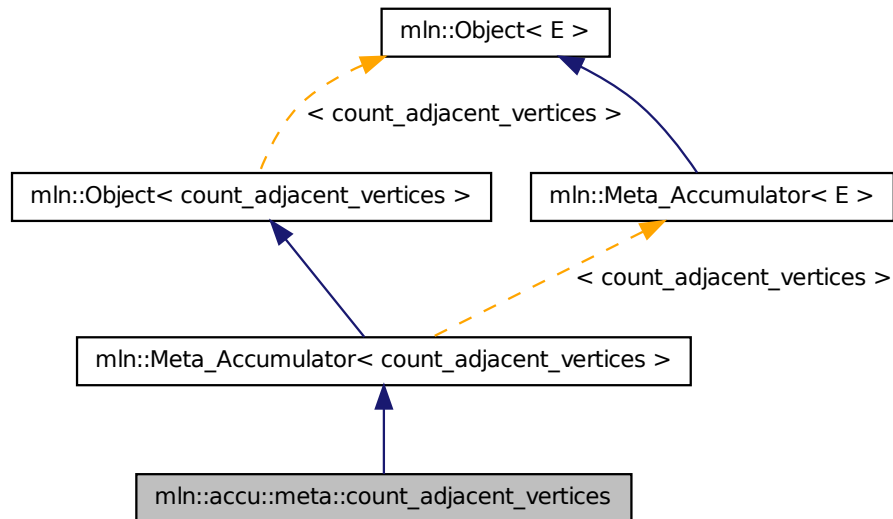
Definition at line 89 of file center.hh.

10.19 mln::accu::meta::count_adjacent_vertices Struct Reference

Meta accumulator for [count_adjacent_vertices](#).

```
#include <count_adjacent_vertices.hh>
```

Inheritance diagram for mln::accu::meta::count_adjacent_vertices:



10.19.1 Detailed Description

Meta accumulator for [count_adjacent_vertices](#).

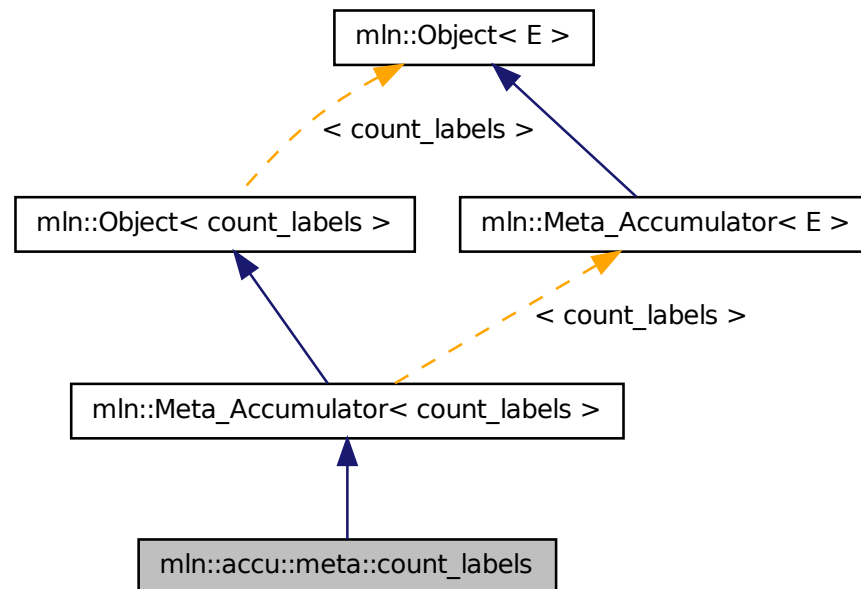
Definition at line 98 of file `accu/count_adjacent_vertices.hh`.

10.20 mln::accu::meta::count_labels Struct Reference

Meta accumulator for [count_labels](#).

```
#include <count_labels.hh>
```

Inheritance diagram for mln::accu::meta::count_labels:



10.20.1 Detailed Description

Meta accumulator for [count_labels](#).

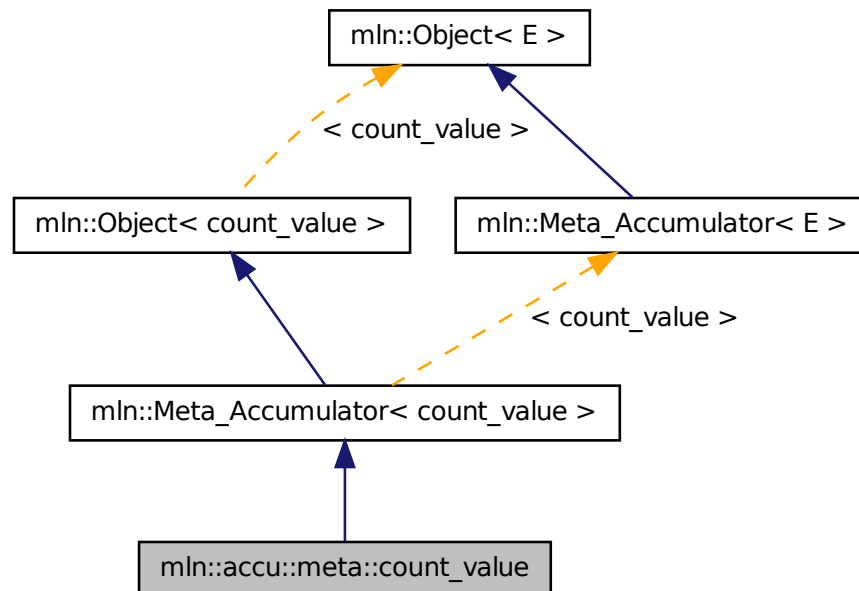
Definition at line 88 of file count_labels.hh.

10.21 mln::accu::meta::count_value Struct Reference

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count_value](#).

```
#include <count_value.hh>
```

Inheritance diagram for mln::accu::meta::count_value:



10.21.1 Detailed Description

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count_value](#).

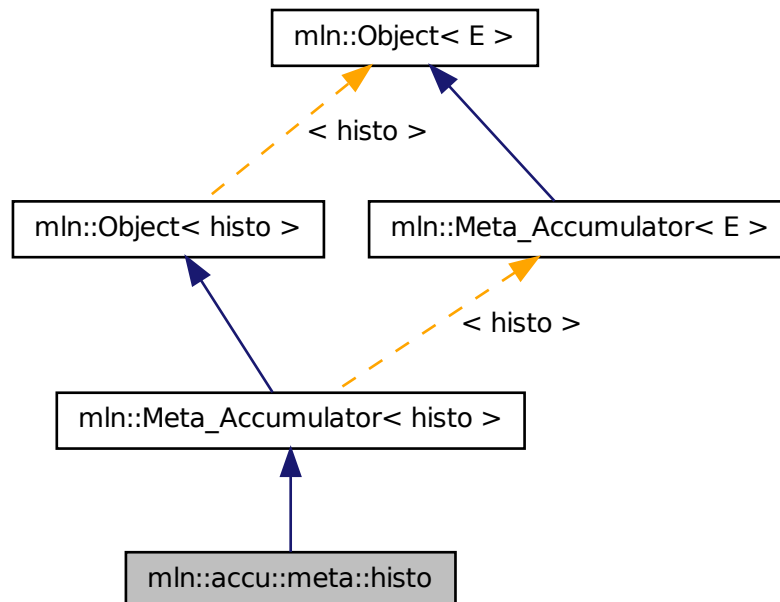
Definition at line 116 of file count_value.hh.

10.22 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

```
#include <histo.hh>
```

Inheritance diagram for mln::accu::meta::histo:



10.22.1 Detailed Description

Meta accumulator for histo.

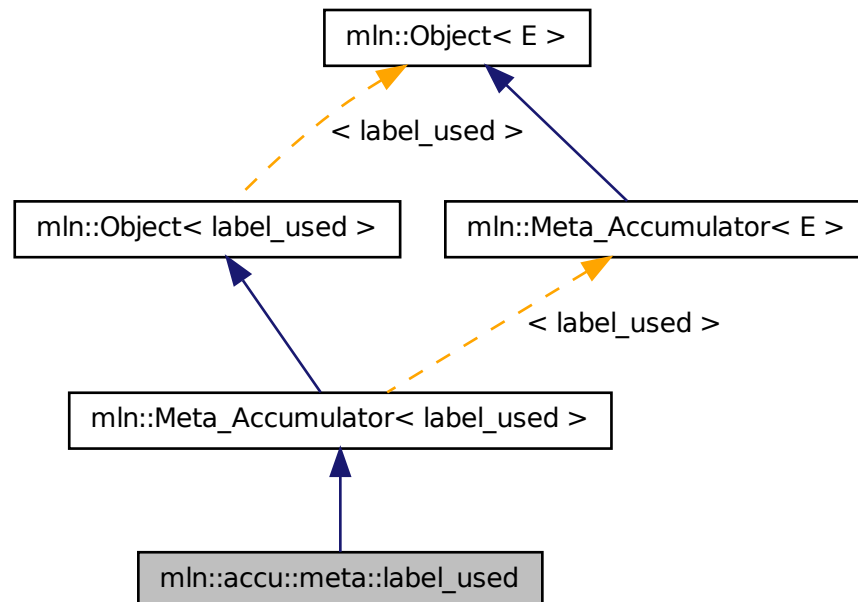
Definition at line 102 of file `accu/histo.hh`.

10.23 mln::accu::meta::label_used Struct Reference

Meta accumulator for [label_used](#).

```
#include <label_used.hh>
```

Inheritance diagram for mln::accu::meta::label_used:



10.23.1 Detailed Description

Meta accumulator for [label_used](#).

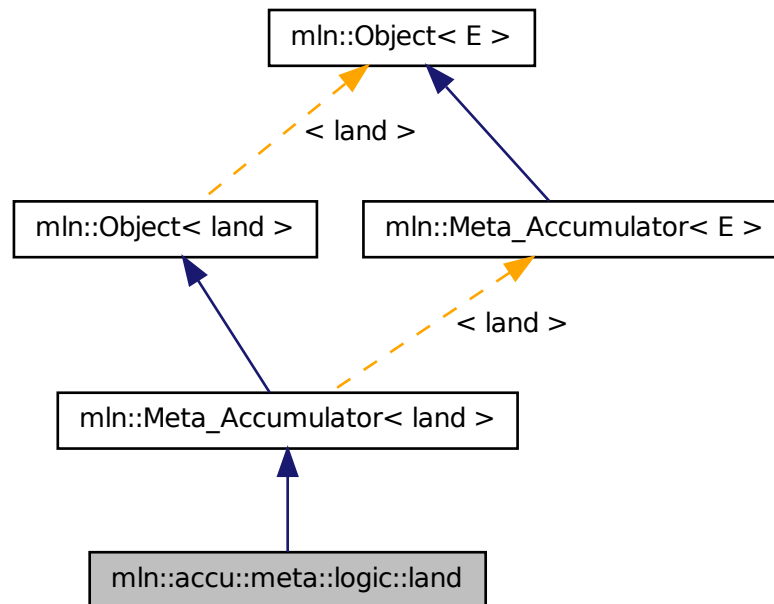
Definition at line 85 of file label_used.hh.

10.24 mln::accu::meta::logic::land Struct Reference

Meta accumulator for land.

```
#include <land.hh>
```


Inheritance diagram for mln::accu::meta::logic::land:



10.24.1 Detailed Description

Meta accumulator for land.

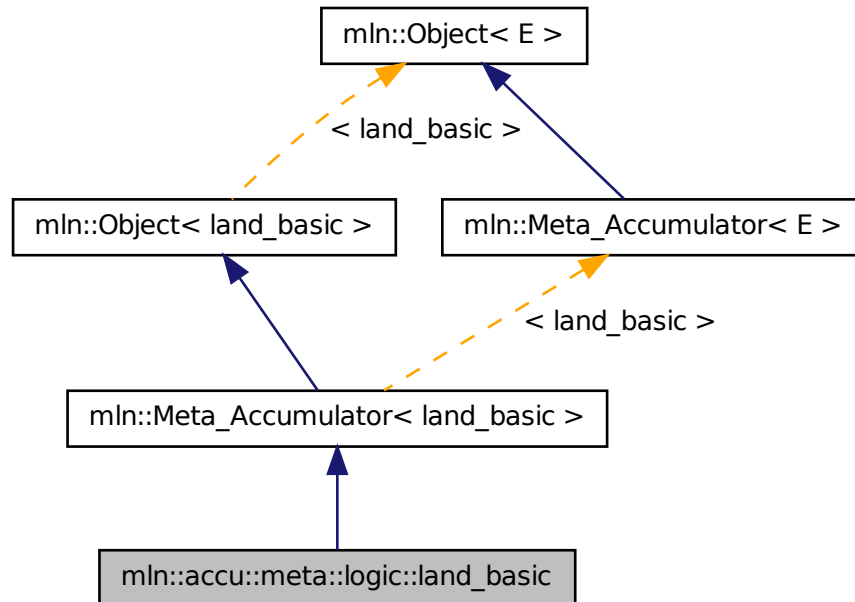
Definition at line 76 of file accu/logic/land.hh.

10.25 mln::accu::meta::logic::land_basic Struct Reference

Meta accumulator for [land_basic](#).

```
#include <land_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::land_basic:



10.25.1 Detailed Description

Meta accumulator for [land_basic](#).

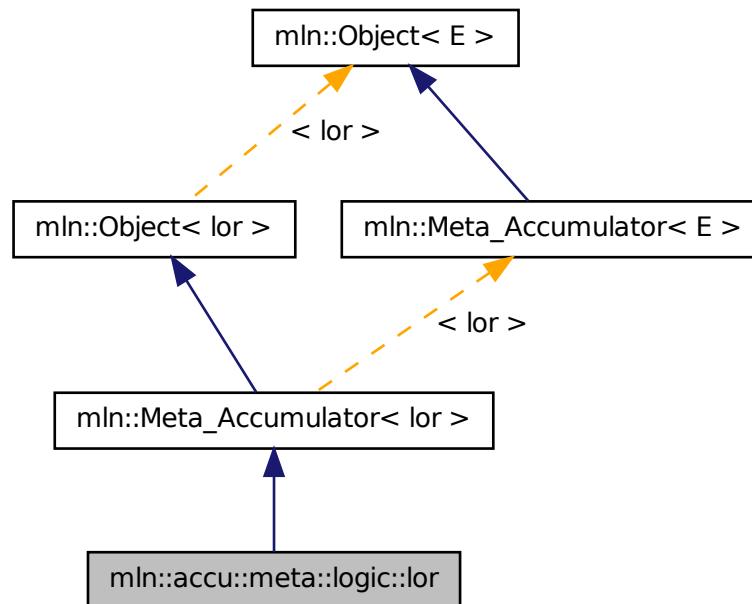
Definition at line 77 of file land_basic.hh.

10.26 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

```
#include <lor.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor:



10.26.1 Detailed Description

Meta accumulator for lor.

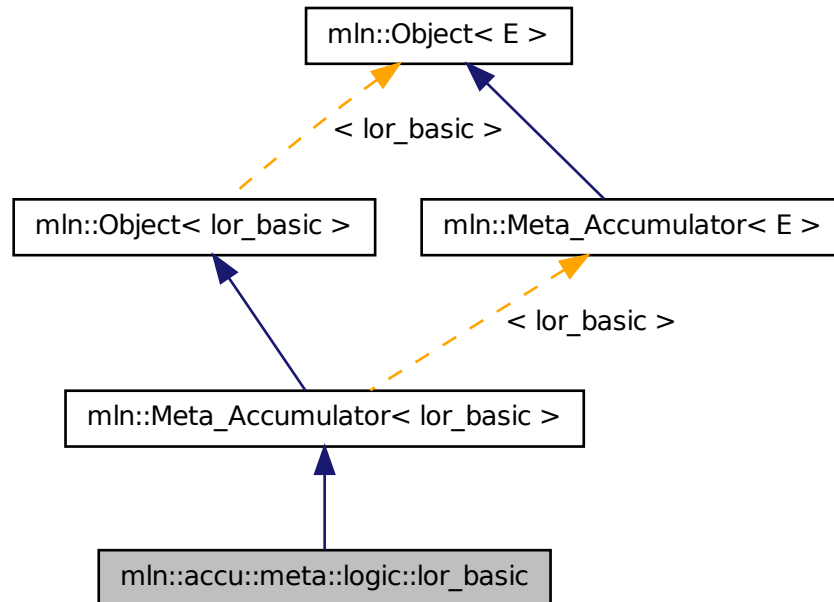
Definition at line 76 of file accu/logic/lor.hh.

10.27 mln::accu::meta::logic::lor_basic Struct Reference

Meta accumulator for [lor_basic](#).

```
#include <lor_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor_basic:



10.27.1 Detailed Description

Meta accumulator for [lor_basic](#).

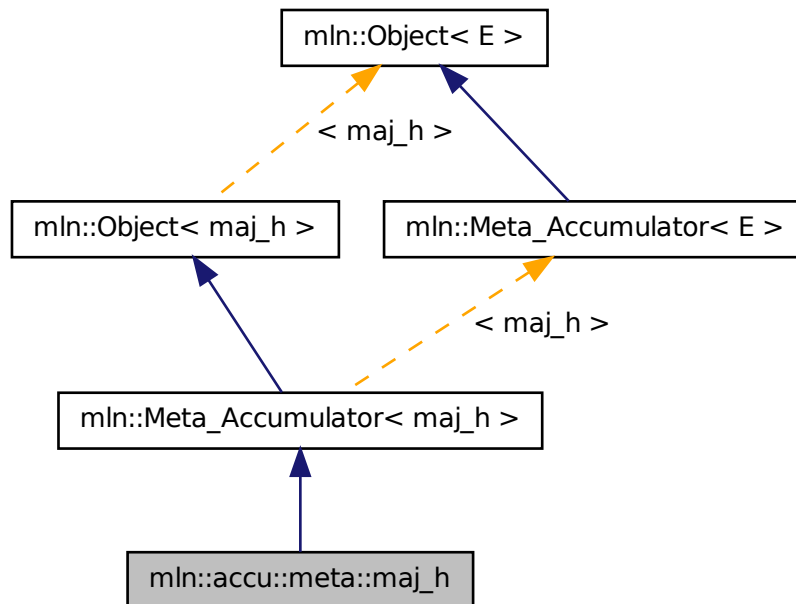
Definition at line 76 of file [lor_basic.hh](#).

10.28 mln::accu::meta::maj_h Struct Reference

Meta accumulator for [maj_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for mln::accu::meta::maj_h:



10.28.1 Detailed Description

Meta accumulator for [maj_h](#).

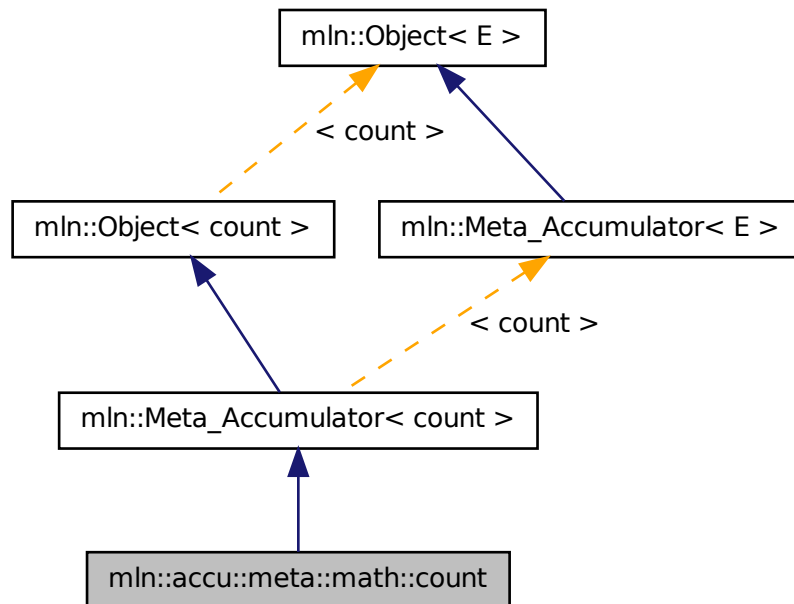
Definition at line 101 of file `maj_h.hh`.

10.29 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

```
#include <count.hh>
```

Inheritance diagram for `mln::accu::meta::math::count`:



10.29.1 Detailed Description

Meta accumulator for count.

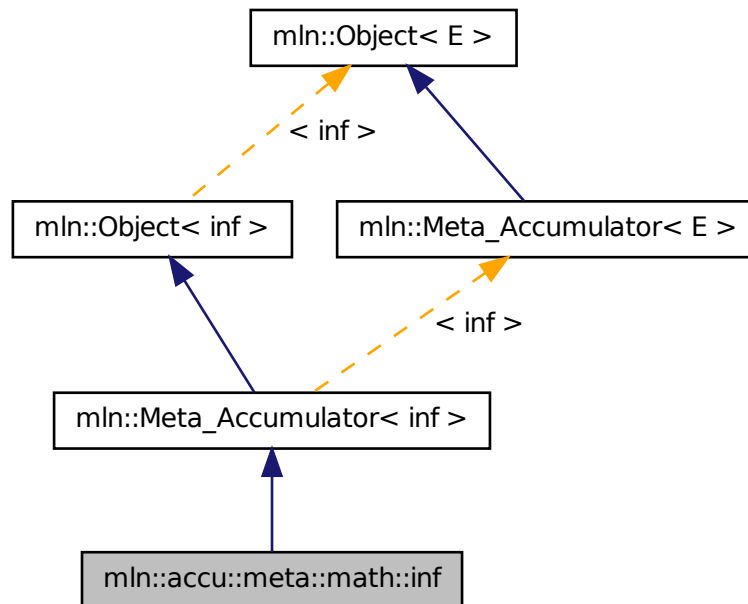
Definition at line 77 of file `count.hh`.

10.30 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

```
#include <inf.hh>
```

Inheritance diagram for mln::accu::meta::math::inf:



10.30.1 Detailed Description

Meta accumulator for inf.

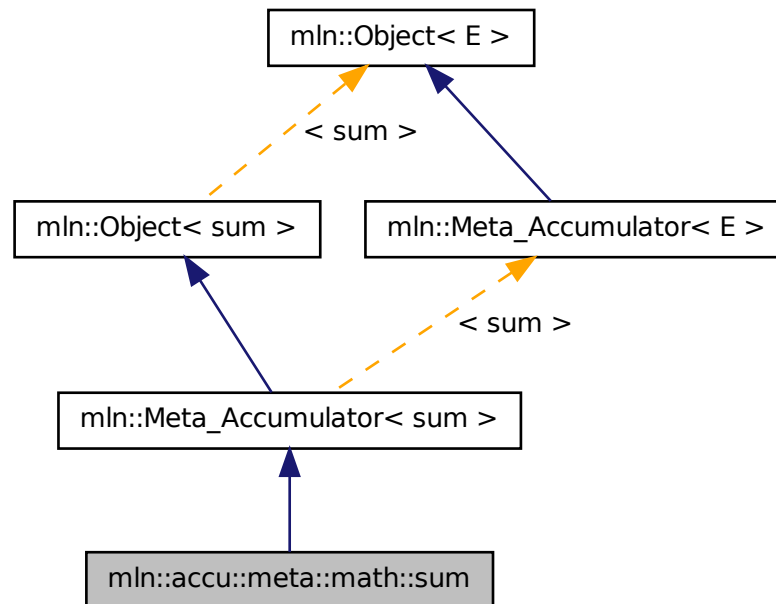
Definition at line 97 of file `accu/math/inf.hh`.

10.31 mln::accu::meta::math::sum Struct Reference

Meta accumulator for sum.

```
#include <sum.hh>
```

Inheritance diagram for mln::accu::meta::math::sum:



10.31.1 Detailed Description

Meta accumulator for sum.

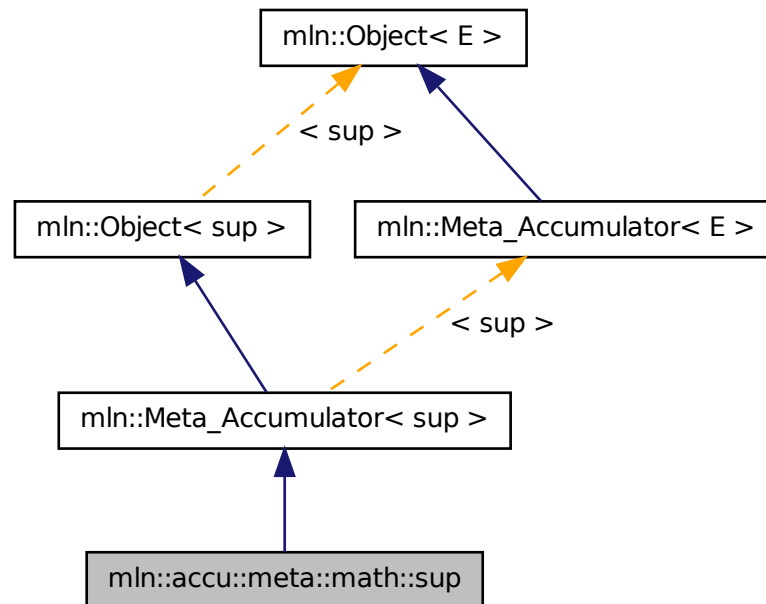
Definition at line 66 of file accu/math/sum.hh.

10.32 mln::accu::meta::math::sup Struct Reference

Meta accumulator for sup.

```
#include <sup.hh>
```


Inheritance diagram for mln::accu::meta::math::sup:



10.32.1 Detailed Description

Meta accumulator for sup.

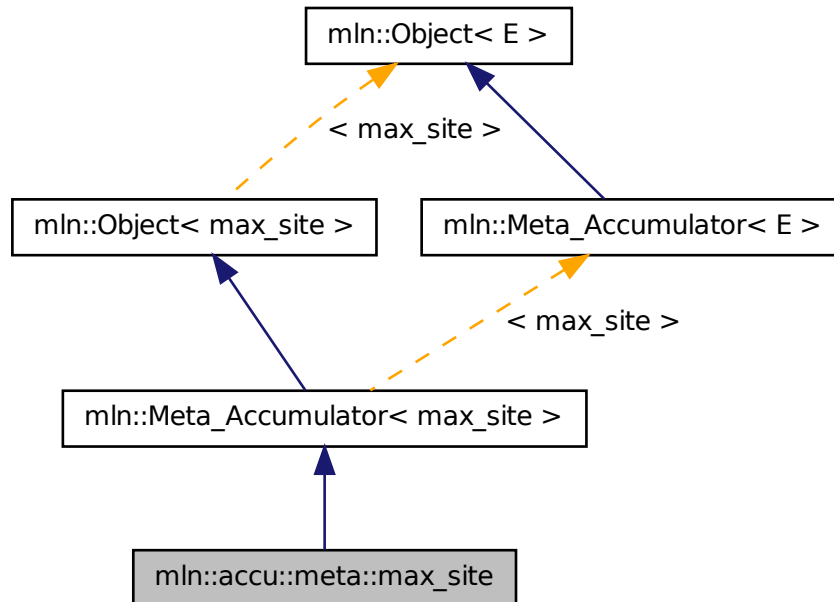
Definition at line 98 of file accu/math/sup.hh.

10.33 mln::accu::meta::max_site Struct Reference

Meta accumulator for [max_site](#).

```
#include <max_site.hh>
```

Inheritance diagram for mln::accu::meta::max_site:



10.33.1 Detailed Description

Meta accumulator for [max_site](#).

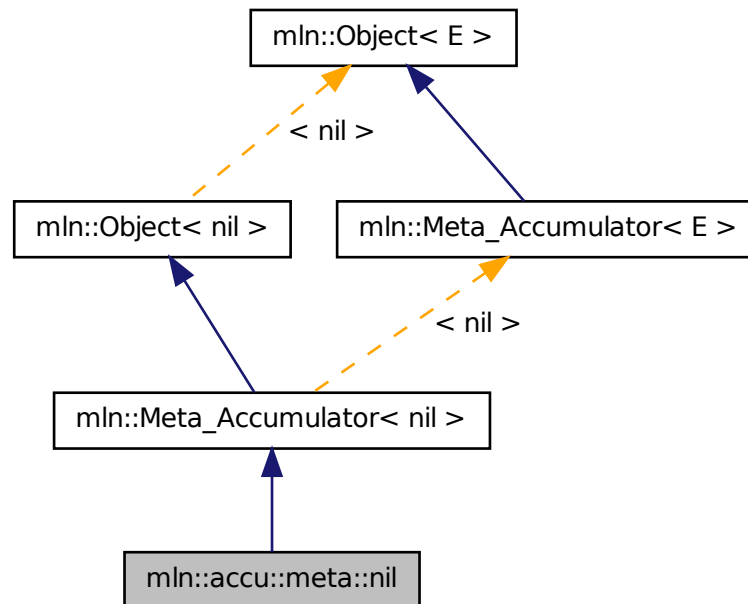
Definition at line 90 of file max_site.hh.

10.34 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

```
#include <nil.hh>
```

Inheritance diagram for mln::accu::meta::nil:



10.34.1 Detailed Description

Meta accumulator for nil.

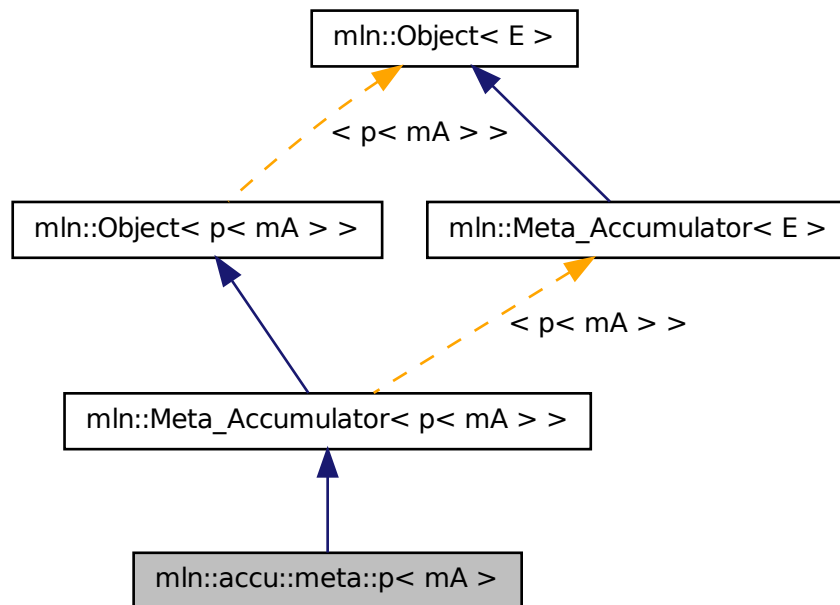
Definition at line 77 of file accu/nil.hh.

10.35 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for p.

```
#include <p.hh>
```

Inheritance diagram for `mln::accu::meta::p< mA >`:



10.35.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::p< mA >
```

Meta accumulator for p.

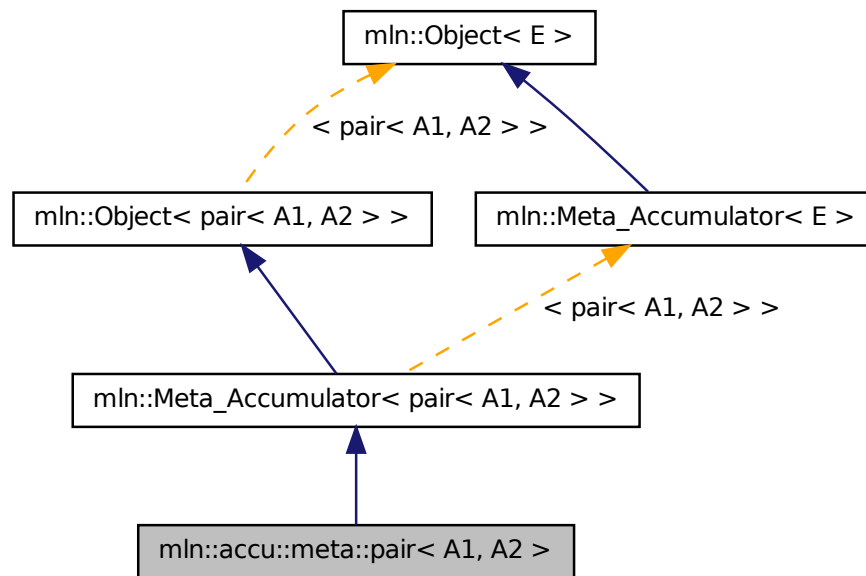
Definition at line 83 of file p.hh.

10.36 mln::accu::meta::pair< A1, A2 > Struct Template Reference

Meta accumulator for pair.

```
#include <pair.hh>
```

Inheritance diagram for mln::accu::meta::pair< A1, A2 >:



10.36.1 Detailed Description

```
template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >
```

Meta accumulator for pair.

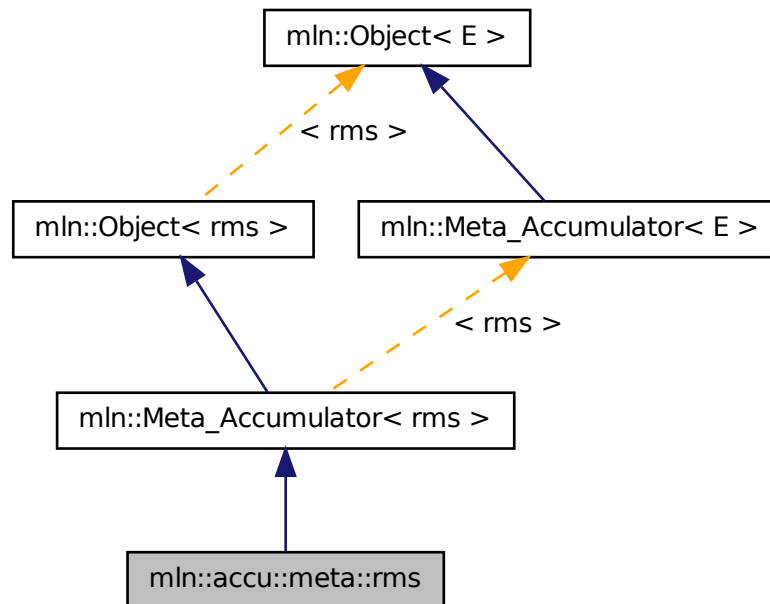
Definition at line 110 of file pair.hh.

10.37 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

```
#include <rms.hh>
```

Inheritance diagram for mln::accu::meta::rms:



10.37.1 Detailed Description

Meta accumulator for rms.

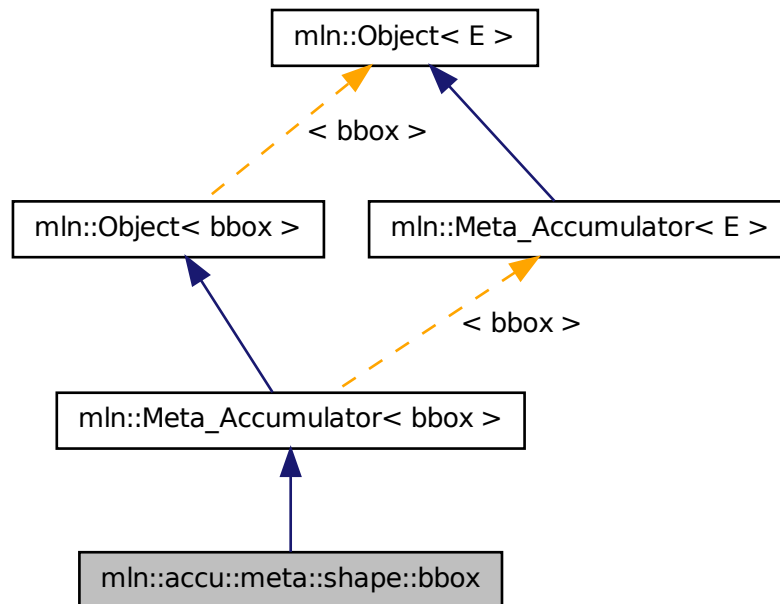
Definition at line 88 of file accu/rms.hh.

10.38 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

```
#include <bbox.hh>
```

Inheritance diagram for mln::accu::meta::shape::bbox:



10.38.1 Detailed Description

Meta accumulator for bbox.

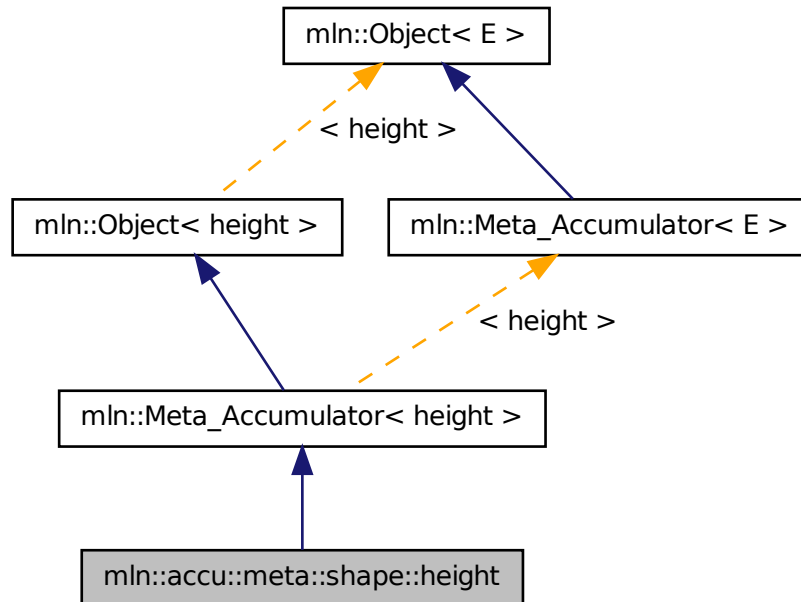
Definition at line 94 of file `accu/shape/bbox.hh`.

10.39 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

```
#include <height.hh>
```

Inheritance diagram for `mln::accu::meta::shape::height`:



10.39.1 Detailed Description

Meta accumulator for height.

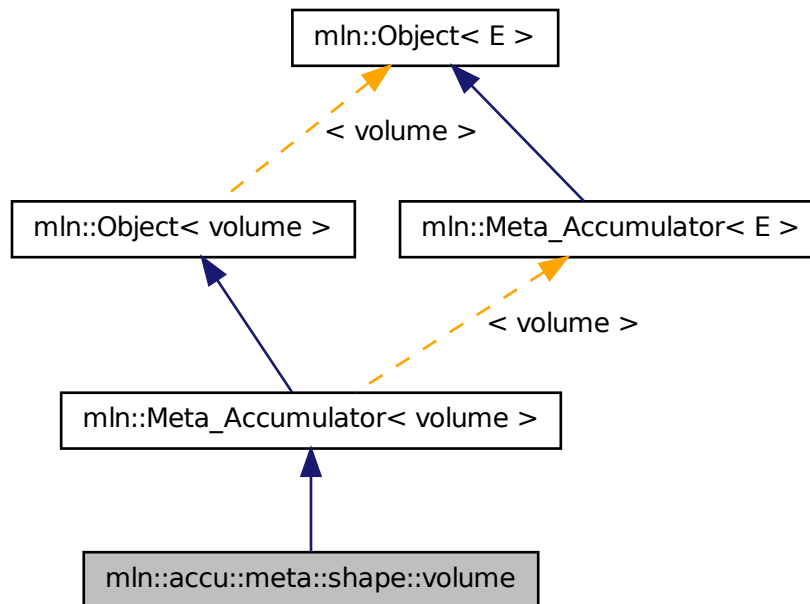
Definition at line 121 of file `accu/shape/height.hh`.

10.40 `mln::accu::meta::shape::volume` Struct Reference

Meta accumulator for volume.

```
#include <volume.hh>
```


Inheritance diagram for mln::accu::meta::shape::volume:



10.40.1 Detailed Description

Meta accumulator for volume.

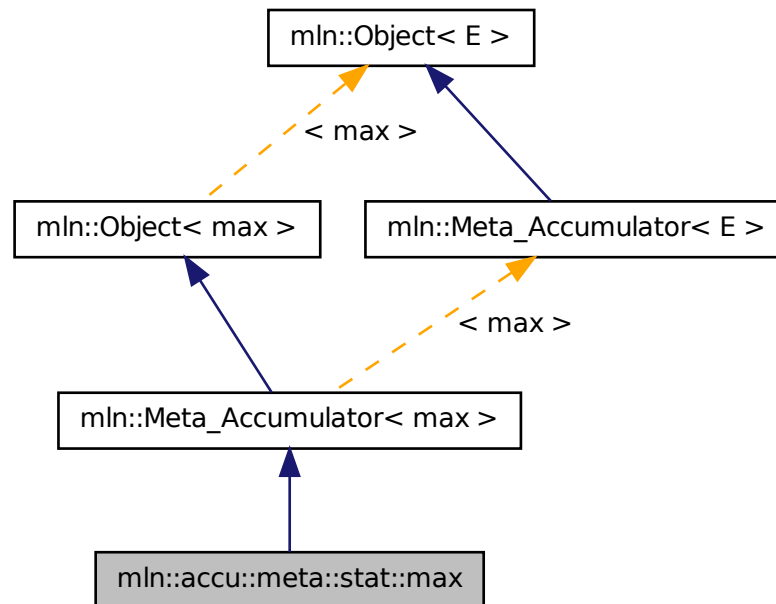
Definition at line 120 of file accu/shape/volume.hh.

10.41 mln::accu::meta::stat::max Struct Reference

Meta accumulator for max.

```
#include <max.hh>
```

Inheritance diagram for mln::accu::meta::stat::max:



10.41.1 Detailed Description

Meta accumulator for max.

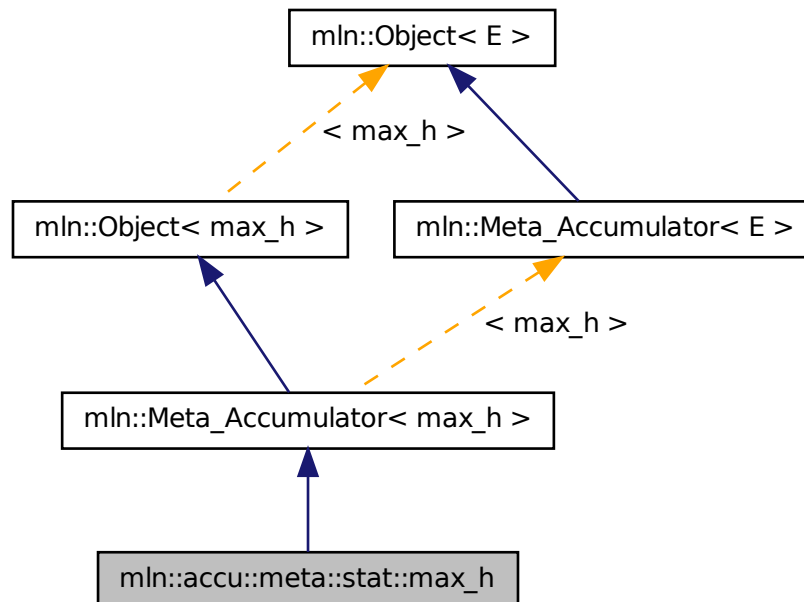
Definition at line 78 of file accu/stat/max.hh.

10.42 mln::accu::meta::stat::max_h Struct Reference

Meta accumulator for max.

```
#include <max_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::max_h:



10.42.1 Detailed Description

Meta accumulator for max.

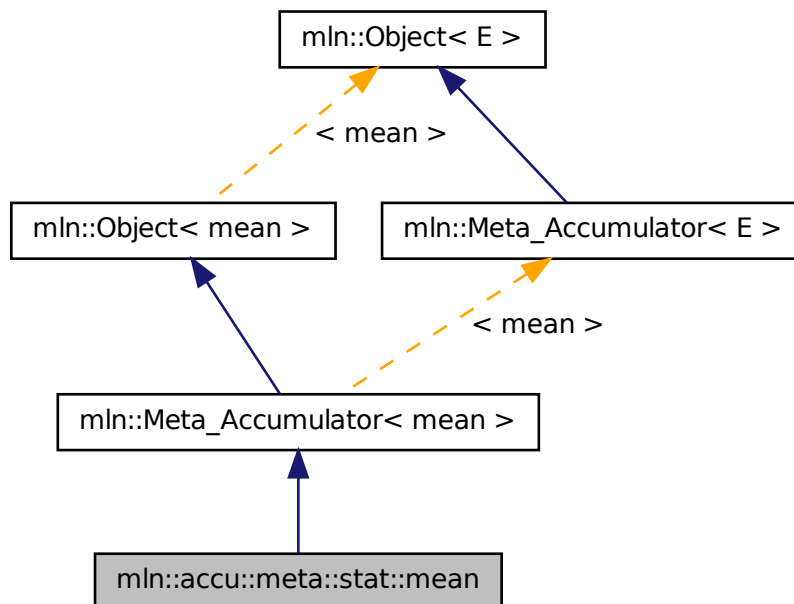
Definition at line 77 of file `max_h.hh`.

10.43 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for mean.

```
#include <mean.hh>
```

Inheritance diagram for mln::accu::meta::stat::mean:



10.43.1 Detailed Description

Meta accumulator for mean.

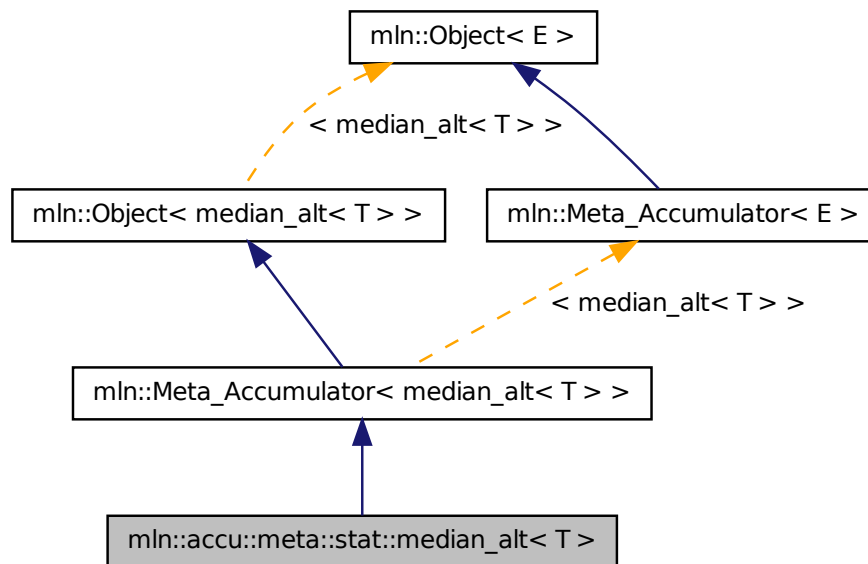
Definition at line 65 of file accu/stat/mean.hh.

10.44 mln::accu::meta::stat::median_alt< T > Struct Template Reference

Meta accumulator for [median_alt](#).

```
#include <median_alt.hh>
```

Inheritance diagram for `mln::accu::meta::stat::median_alt< T >`:



10.44.1 Detailed Description

`template<typename T> struct mln::accu::meta::stat::median_alt< T >`

Meta accumulator for [median_alt](#).

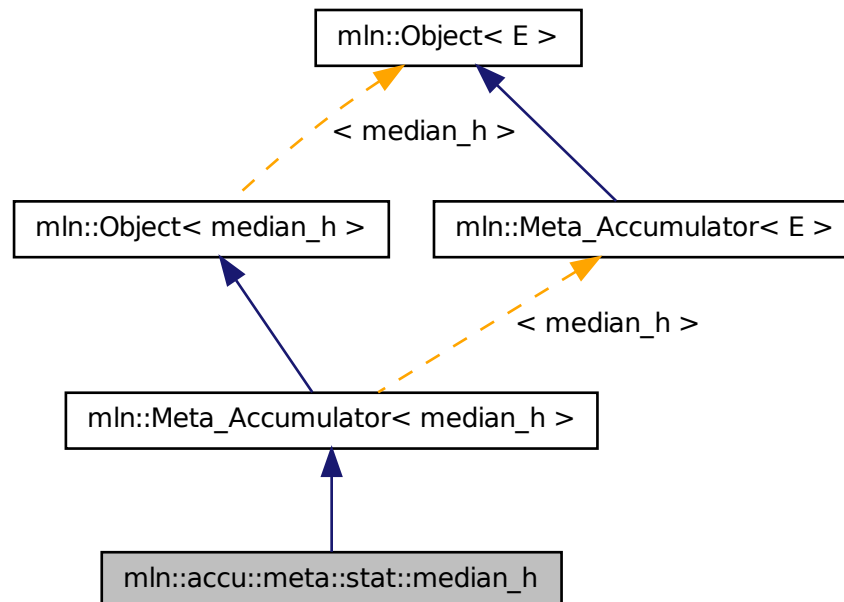
Definition at line 122 of file `median_alt.hh`.

10.45 mln::accu::meta::stat::median_h Struct Reference

Meta accumulator for [median_h](#).

`#include <median_h.hh>`

Inheritance diagram for mln::accu::meta::stat::median_h:



10.45.1 Detailed Description

Meta accumulator for [median_h](#).

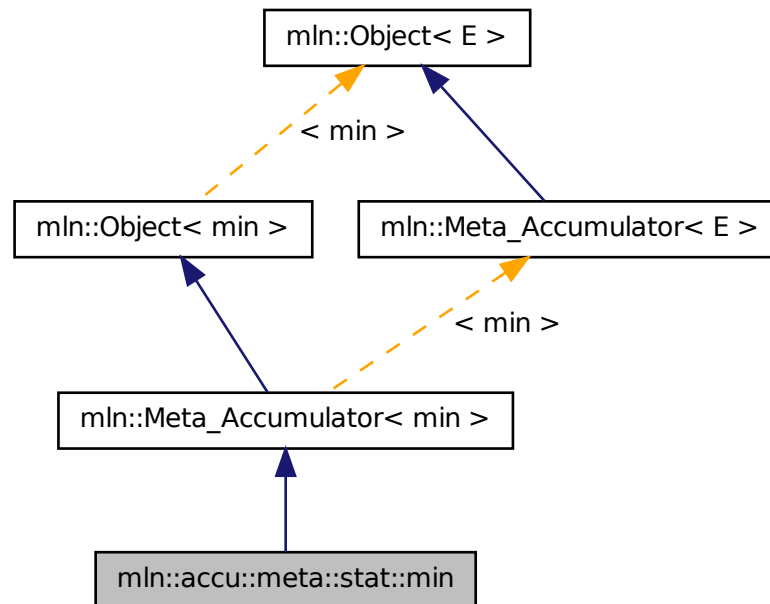
Definition at line 61 of file median_h.hh.

10.46 mln::accu::meta::stat::min Struct Reference

Meta accumulator for min.

```
#include <min.hh>
```

Inheritance diagram for mln::accu::meta::stat::min:



10.46.1 Detailed Description

Meta accumulator for min.

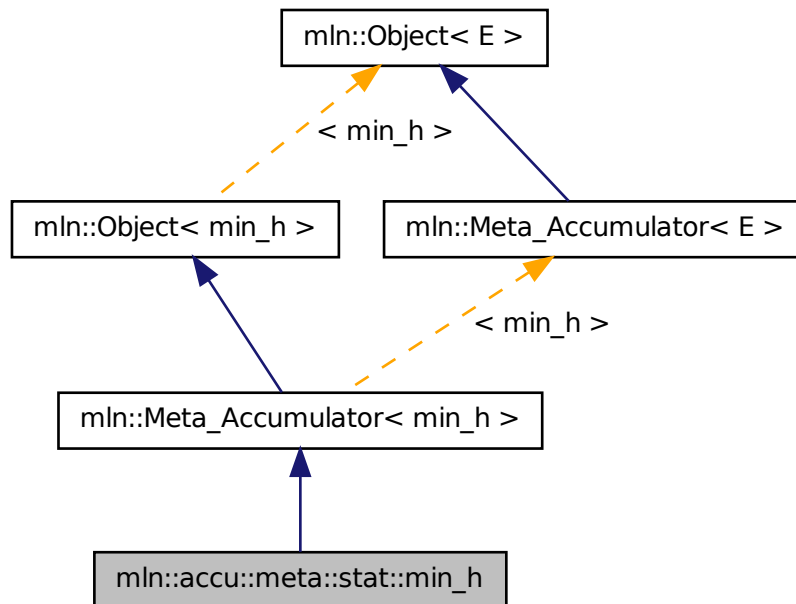
Definition at line 78 of file accu/stat/min.hh.

10.47 mln::accu::meta::stat::min_h Struct Reference

Meta accumulator for min.

```
#include <min_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::min_h:



10.47.1 Detailed Description

Meta accumulator for min.

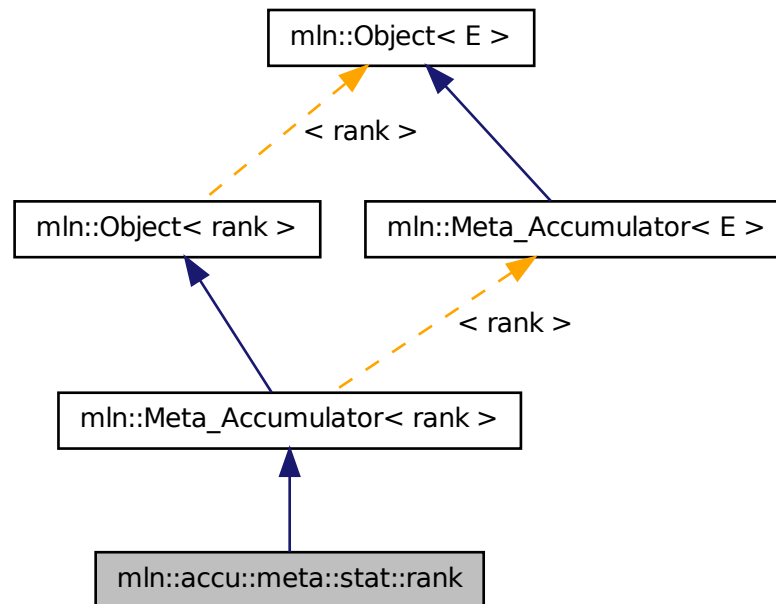
Definition at line 77 of file min_h.hh.

10.48 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

```
#include <rank.hh>
```


Inheritance diagram for mln::accu::meta::stat::rank:



10.48.1 Detailed Description

Meta accumulator for rank.

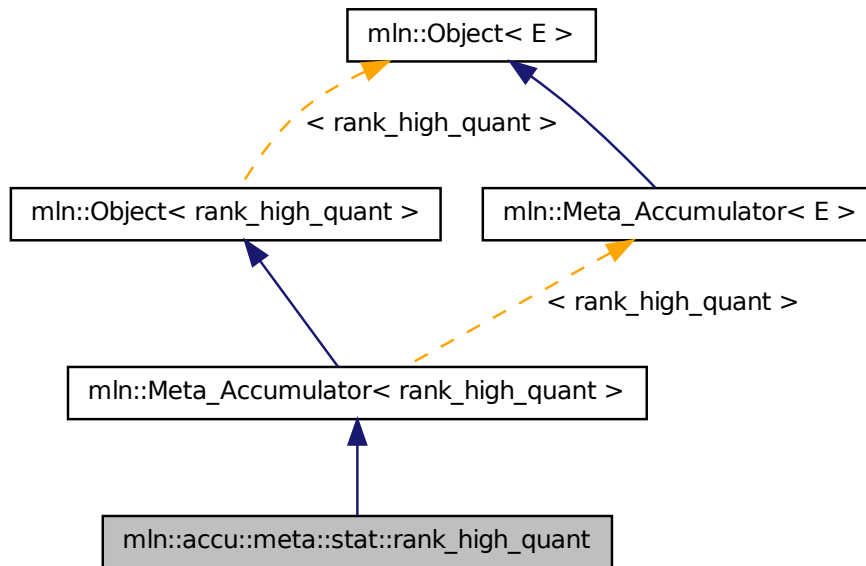
Definition at line 123 of file rank.hh.

10.49 mln::accu::meta::stat::rank_high_quant Struct Reference

Meta accumulator for [rank_high_quant](#).

```
#include <rank_high_quant.hh>
```

Inheritance diagram for `mln::accu::meta::stat::rank_high_quant`:



10.49.1 Detailed Description

Meta accumulator for [rank_high_quant](#).

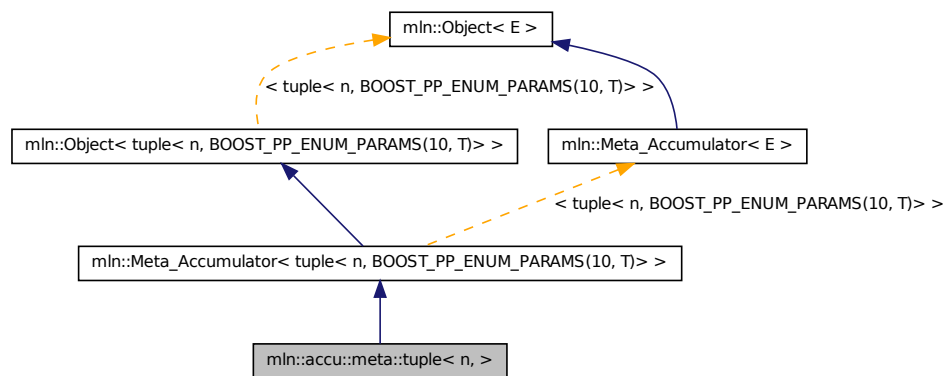
Definition at line 102 of file `rank_high_quant.hh`.

10.50 `mln::accu::meta::tuple< n, >` Struct Template Reference

Meta accumulator for tuple.

```
#include <tuple.hh>
```

Inheritance diagram for mln::accu::meta::tuple< n, >:



10.50.1 Detailed Description

```
template<unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T,
boost::tuples::null_type)> struct mln::accu::meta::tuple< n, >
```

Meta accumulator for tuple.

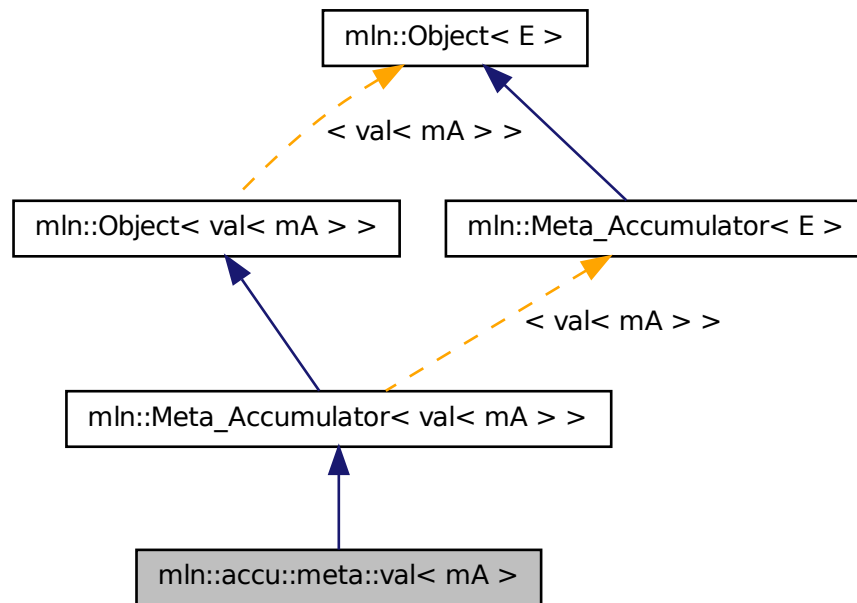
Definition at line 111 of file tuple.hh.

10.51 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for val.

```
#include <v.hh>
```

Inheritance diagram for `mln::accu::meta::val< mA >`:



10.51.1 Detailed Description

`template<typename mA> struct mln::accu::meta::val< mA >`

Meta accumulator for val.

Definition at line 87 of file v.hh.

10.52 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits base< util::ignore, nil< T > >.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)

Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t .

- [util::ignore to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

10.52.1 Detailed Description

template<typename T> struct mln::accu::nil< T >

Define an accumulator that does nothing.

Definition at line 49 of file accu/nil.hh.

10.52.2 Member Function Documentation

10.52.2.1 template<typename T> void mln::accu::nil< T >::init () [inline]

Manipulators.

Definition at line 100 of file accu/nil.hh.

10.52.2.2 template<typename T> bool mln::accu::nil< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 136 of file accu/nil.hh.

10.52.2.3 void mln::Accumulator< nil< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.52.2.4 void mln::Accumulator< nil< T > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.52.2.5 `template<typename T> util::ignore mln::accu::nil< T>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 128 of file `accu/nil.hh`.

10.53 `mln::accu::p< A >` Struct Template Reference

Generic p of accumulators.

`#include <p.hh>`

Inherits `base< const A::result &, p< A >>`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const A::result & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.53.1 Detailed Description

`template<typename A> struct mln::accu::p< A >`

Generic p of accumulators. The parameter V is the type of values.

Definition at line 50 of file `p.hh`.

10.53.2 Member Function Documentation

10.53.2.1 `template<typename A> void mln::accu::p< A>::init () [inline]`

Manipulators.

Definition at line 115 of file `p.hh`.

10.53.2.2 `template<typename A> bool mln::accu::p<A>::is_valid() const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 155 of file p.hh.

10.53.2.3 `void mln::Accumulator< p< A > >::take_as_init(const T & t) [inherited]`

Take as initialization the value τ .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

```
10.53.2.4 void mln::Accumulator< p< A > >::take_n_times ( unsigned n, const T & t )
[inherited]
```

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

```
10.53.2.5 template<typename A> const A::result & mln::accu::p< A >::to_result ( ) const
[inline]
```

Get the value of the accumulator.

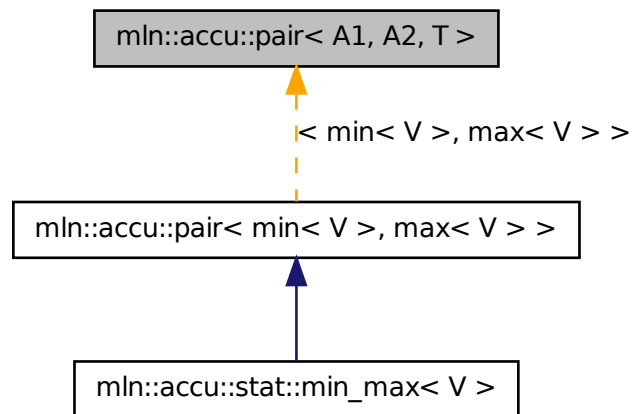
Definition at line 147 of file p.hh.

10.54 mln::accu::pair< A1, A2, T > Struct Template Reference

Generic pair of accumulators.

```
#include <pair.hh>
```

Inheritance diagram for `mln::accu::pair< A1, A2, T >`:



Public Member Functions

- `A1::result first () const`
Return the result of the first accumulator.
- `A1 first_accu () const`
Return the first accumulator.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `A2::result second () const`
Return the result of the second accumulator.
- `A2 second_accu () const`
Return the second accumulator.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `void init ()`
Manipulators.

- `std::pair< typename A1::result, typename A2::result > to_result () const`
Get the value of the accumulator.

10.54.1 Detailed Description

template<typename A1, typename A2, typename T = mln_argument(A1)> struct mln::accu::pair< A1, A2, T >

Generic pair of accumulators. The parameter T is the type of values.

Definition at line 58 of file pair.hh.

10.54.2 Member Function Documentation

10.54.2.1 template<typename A1 , typename A2 , typename T > A1::result mln::accu::pair< A1, A2, T >::first () const [inline]

Return the result of the first accumulator.

Definition at line 191 of file pair.hh.

10.54.2.2 template<typename A1 , typename A2 , typename T > A1 mln::accu::pair< A1, A2, T >::first_accu () const [inline]

Return the first accumulator.

Definition at line 209 of file pair.hh.

10.54.2.3 template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init () [inline]

Manipulators.

Definition at line 136 of file pair.hh.

10.54.2.4 template<typename A1 , typename A2 , typename T > bool mln::accu::pair< A1, A2, T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 226 of file pair.hh.

10.54.2.5 template<typename A1 , typename A2 , typename T > A2::result mln::accu::pair< A1, A2, T >::second () const [inline]

Return the result of the second accumulator.

Definition at line 199 of file pair.hh.

10.54.2.6 `template<typename A1 , typename A2 , typename T > A2 mln::accu::pair< A1, A2, T >::second_accu () const [inline]`

Return the second accumulator.

Definition at line 217 of file pair.hh.

10.54.2.7 `void mln::Accumulator< pair< A1, A2, T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.54.2.8 `void mln::Accumulator< pair< A1, A2, T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.54.2.9 `template<typename A1 , typename A2 , typename T > std::pair< typename A1::result, typename A2::result > mln::accu::pair< A1, A2, T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 172 of file pair.hh.

10.55 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits base< V, rms< T, V > >.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
*Take as initialization the value *t*.*
- `void take_n_times (unsigned n, const T &t)`
*Take *n* times the value *t*.*
- `V to_result () const`
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.55.1 Detailed Description

template<typename T, typename V> struct mln::accu::rms< T, V >

Generic root mean square accumulator class. The parameter T is the type of the root mean square value.

Definition at line 52 of file accu/rms.hh.

10.55.2 Member Function Documentation

**10.55.2.1 template<typename T , typename V > void mln::accu::rms< T, V >::init ()
 [inline]**

Manipulators.

Definition at line 112 of file accu/rms.hh.

References mln::literal::zero.

**10.55.2.2 template<typename T , typename V > bool mln::accu::rms< T, V >::is_valid () const
 [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 166 of file accu/rms.hh.

10.55.2.3 void mln::Accumulator< rms< T, V > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

**10.55.2.4 void mln::Accumulator< rms< T, V > >::take_n_times (unsigned n, const T & t)
 [inherited]**

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

**10.55.2.5 template<typename T , typename V > V mln::accu::rms< T, V >::to_result () const
 [inline]**

Get the value of the accumulator.

Definition at line 148 of file accu/rms.hh.

10.56 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding box accumulator class.

```
#include <bbox.hh>
```

Inherits base< const box< P > &, bbox< P > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const [box](#)< P > & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.56.1 Detailed Description

template<typename P> struct mln::accu::shape::bbox< P >

Generic bounding box accumulator class. The parameter P is the type of points.

Definition at line 55 of file accu/shape/bbox.hh.

10.56.2 Member Function Documentation

10.56.2.1 template<typename P> void mln::accu::shape::bbox< P >::init () [inline]

Manipulators.

Definition at line 124 of file accu/shape/bbox.hh.

10.56.2.2 template<typename P> bool mln::accu::shape::bbox< P >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 224 of file accu/shape/bbox.hh.

10.56.2.3 void mln::Accumulator< bbox< P > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.56.2.4 void mln::Accumulator< bbox< P > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.56.2.5 template<typename P> const box< P > & mln::accu::shape::bbox< P >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 215 of file accu/shape/bbox.hh.

Referenced by mln::geom::rotate().

10.57 mln::accu::shape::height< I > Struct Template Reference

Height accumulator.

```
#include <height.hh>
```

Inherits base< unsigned, height< I > >.

Public Types

- typedef [util::pix< I >](#) [argument](#)
The accumulated data type.
- typedef [argument::value](#) [value](#)
The value type associated to the pixel type.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- unsigned [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned h)
Force the value of the counter to h.

10.57.1 Detailed Description

template<typename I> struct mln::accu::shape::height< I >

Height accumulator. The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 68 of file accu/shape/height.hh.

10.57.2 Member Typedef Documentation

10.57.2.1 template<typename I> typedef util::pix<I> mln::accu::shape::height< I >::argument

The accumulated data type.

The height of component is represented by the height of its root pixel. See [mln::morpho::closing_height](#) and [mln::morpho::opening_height](#) for actual uses of this accumulator. **FIXME:** Replaced by [mln::morpho::attribute::height](#)

Definition at line 78 of file accu/shape/height.hh.

10.57.2.2 template<typename I> typedef argument::value mln::accu::shape::height< I >::value

The value type associated to the pixel type.

Definition at line 80 of file accu/shape/height.hh.

10.57.3 Member Function Documentation

10.57.3.1 template<typename I> void mln::accu::shape::height< I >::init () [inline]

Manipulators.

Definition at line 150 of file accu/shape/height.hh.

10.57.3.2 template<typename I> bool mln::accu::shape::height< I >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 199 of file accu/shape/height.hh.

10.57.3.3 `template<typename I> void mln::accu::shape::height< I >::set_value (unsigned h)`
`[inline]`

Force the value of the counter to h .

Definition at line 188 of file accu/shape/height.hh.

10.57.3.4 `void mln::Accumulator< height< I > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.57.3.5 `void mln::Accumulator< height< I > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.57.3.6 `template<typename I> unsigned mln::accu::shape::height< I >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 180 of file accu/shape/height.hh.

10.58 mln::accu::shape::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits `base< unsigned, volume< I > >`.

Public Types

- typedef `util::pix< I >` `argument`
The accumulated data type.
- typedef `argument::value` `value`
The value type associated to the pixel type.

Public Member Functions

- bool `is_valid () const`
Check whether this accu is able to return a result.
- void `take_as_init (const T &t)`
Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t.

- unsigned [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

- void [set_value](#) (unsigned v)

Force the value of the counter to v.

10.58.1 Detailed Description

template<typename I> struct mln::accu::shape::volume< I >

Volume accumulator class. The parameter I is the image type on which the accumulator of pixels is built. Definition at line 66 of file accu/shape/volume.hh.

10.58.2 Member Typedef Documentation

10.58.2.1 template<typename I> typedef util::pix<I> mln::accu::shape::volume< I >::argument

The accumulated data type.

The volume of component is represented by the volume of its root pixel. See mln::morpho::closing_volume and mln::morpho::opening_volume for actual uses of this accumulator. FIXME: Replaced by [mln::morpho::attribute::volume](#)

Definition at line 76 of file accu/shape/volume.hh.

10.58.2.2 template<typename I> typedef argument::value mln::accu::shape::volume< I >::value

The value type associated to the pixel type.

Definition at line 78 of file accu/shape/volume.hh.

10.58.3 Member Function Documentation

10.58.3.1 template<typename I> void mln::accu::shape::volume< I >::init () [inline]

Manipulators.

Definition at line 148 of file accu/shape/volume.hh.

References mln::literal::zero.

10.58.3.2 `template<typename I> bool mln::accu::shape::volume< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 215 of file accu/shape/volume.hh.

10.58.3.3 `template<typename I> void mln::accu::shape::volume< I >::set_value (unsigned v) [inline]`

Force the value of the counter to *v*.

Definition at line 204 of file accu/shape/volume.hh.

References mln::literal::zero.

10.58.3.4 `void mln::Accumulator< volume< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.58.3.5 `void mln::Accumulator< volume< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.58.3.6 `template<typename I> unsigned mln::accu::shape::volume< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 196 of file accu/shape/volume.hh.

10.59 mln::accu::site_set::rectangularity< P > Class Template Reference

Compute the rectangularity of a site set.

```
#include <rectangularity.hh>
```

Inherits couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >.

Public Member Functions

- A2::result [area](#) () const

Return the site set area.

- `A1::result bbox () const`
Return the site set bounding box.
- `rectangularity ()`
Constructor.
- `template<typename T >`
`void take_as_init (const T &t)`
Take as initialization the value t .
- `template<typename T >`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `result to_result () const`
Return the rectangularity value.

10.59.1 Detailed Description

`template<typename P> class mln::accu::site_set::rectangularity< P >`

Compute the rectangularity of a site set.

Definition at line 51 of file `rectangularity.hh`.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 `template<typename P > mln::accu::site_set::rectangularity< P >::rectangularity ()`
`[inline]`

Constructor.

Definition at line 91 of file `rectangularity.hh`.

10.59.3 Member Function Documentation

10.59.3.1 `template<typename P > rectangularity< P >::A2::result`
`mln::accu::site_set::rectangularity< P >::area () const [inline]`

Return the site set area.

Definition at line 107 of file `rectangularity.hh`.

10.59.3.2 `template<typename P > rectangularity< P >::A1::result`
`mln::accu::site_set::rectangularity< P >::bbox () const [inline]`

Return the site set bounding box.

Definition at line 98 of file `rectangularity.hh`.

10.59.3.3 `template<typename E> template<typename T> void mln::Accumulator< E>::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.59.3.4 `template<typename E> template<typename T> void mln::Accumulator< E>::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.59.3.5 `template<typename P> rectangularity< P>::result mln::accu::site_set::rectangularity< P>::to_result () const [inline]`

Return the rectangularity value.

Definition at line 116 of file `rectangularity.hh`.

10.60 mln::accu::stat::deviation< T, S, M > Struct Template Reference

Generic standard deviation accumulator class.

```
#include <deviation.hh>
```

Inherits `base< M, deviation< T, S, M > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `M to_result () const`
Get the value of the accumulator.

- void `init()`
Manipulators.

10.60.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>
struct mln::accu::stat::deviation< T, S, M >**

Generic standard deviation accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the standard deviation; the default type of S is the summation type (property) of T. Parameter M is the type of the mean value; the default type of M is S.

Definition at line 62 of file deviation.hh.

10.60.2 Member Function Documentation

10.60.2.1 template<typename T, typename S, typename M > void mln::accu::stat::deviation< T, S, M >::init() [inline]

Manipulators.

Definition at line 132 of file deviation.hh.

10.60.2.2 template<typename T, typename S, typename M > bool mln::accu::stat::deviation< T, S, M >::is_valid() const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file deviation.hh.

10.60.2.3 void mln::Accumulator< deviation< T, S, M > >::take_as_init(const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.60.2.4 void mln::Accumulator< deviation< T, S, M > >::take_n_times(unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.60.2.5 template<typename T, typename S, typename M > M mln::accu::stat::deviation< T, S, M >::to_result() const [inline]

Get the value of the accumulator.

Definition at line 159 of file deviation.hh.

10.61 mln::accu::stat::histo3d_rgb< V > Struct Template Reference

Define a histogram as accumulator which returns an [image3d](#).

```
#include <histo3d_rgb.hh>
```

Inherits base< [image3d](#)< unsigned >, [histo3d_rgb](#)< V > >.

Public Member Functions

- [bool](#) [is_valid](#) () const
Check whether this accumulator is able to return a result.
- [void](#) [take_as_init](#) (const T &t)
Take as initialization the value t.
- [void](#) [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- [histo3d_rgb](#) ()
Constructors.
- [void](#) [init](#) ()
Manipulators.
- [void](#) [take](#) (const argument &t)
Update the histogram with the RGB pixel t.
- [void](#) [take](#) (const [histo3d_rgb](#)< V > &other)
Update the histogram with an other histogram.
- [result to_result](#) () const
Accessors.

10.61.1 Detailed Description

```
template<typename V> struct mln::accu::stat::histo3d_rgb< V >
```

Define a histogram as accumulator which returns an [image3d](#). Param V defines the type of the input image value. It is in this space that we count the values. For instance, this histogram works well for [image2d](#)< [rgb](#)<2> > or with [image2d](#)< [rgb](#)<7> >. The number of bins depends directly the values V. For 8 bits there is 256x3 bins. Note that less quantification works too.

Definition at line 167 of file [histo3d_rgb.hh](#).

10.61.2 Constructor & Destructor Documentation

10.61.2.1 `template<typename V> mln::accu::stat::histo3d_rgb< V >::histo3d_rgb ()`
`[inline]`

Constructors.

Infer the size of the resulting [image3d](#) domain. By evaluating the minimum and the maximum of V, we define the domain of the resulting [image3d](#).

Definition at line 244 of file `histo3d_rgb.hh`.

10.61.3 Member Function Documentation

10.61.3.1 `template<typename V> void mln::accu::stat::histo3d_rgb< V >::init ()` `[inline]`

Manipulators.

Initialize the histogram with zero value. This method must be called just before starting the use of the histogram. If it's not, resulting values won't converge to the density.

Definition at line 268 of file `histo3d_rgb.hh`.

References `mln::data::fill()`, and `mln::literal::zero`.

10.61.3.2 `template<typename V> bool mln::accu::stat::histo3d_rgb< V >::is_valid () const`
`[inline]`

Check whether this accumulator is able to return a result.

Depends if the resulting [image1d](#) is valid. We can assume it is quite always the case.

Definition at line 307 of file `histo3d_rgb.hh`.

10.61.3.3 `template<typename V> void mln::accu::stat::histo3d_rgb< V >::take (const`
`argument & t)` `[inline]`

Update the histogram with the RGB pixel `t`.

Parameters

`[in]` `t` a graylevel pixel of type V.

The end user shouldn't call this method. In place of it, he can go through the data compute interface.

Definition at line 275 of file `histo3d_rgb.hh`.

10.61.3.4 `template<typename V> void mln::accu::stat::histo3d_rgb< V >::take (const`
`histo3d_rgb< V > & other)` `[inline]`

Update the histogram with an other histogram.

Parameters

`[in]` `other` the other histogram.

The end user shouldn't call this method. This is part of data compute interface mechanism.

Definition at line 286 of file histo3d_rgb.hh.

10.61.3.5 `void mln::Accumulator< histo3d_rgb< V > >::take_as_init (const T & t)`
[**inherited**]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.61.3.6 `void mln::Accumulator< histo3d_rgb< V > >::take_n_times (unsigned n, const T & t)` [**inherited**]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.61.3.7 `template<typename V> histo3d_rgb< V >::result mln::accu::stat::histo3d_rgb< V >::to_result () const` [**inline**]

Accessors.

Return the histogram as an RGB [image3d](#). This is the machinery to communicate with data compute interface. The end user should'nt use it.

Definition at line 293 of file histo3d_rgb.hh.

10.62 mln::accu::stat::max< T > Struct Template Reference

Generic max accumulator class.

```
#include <max.hh>
```

Inherits base< const T &, max< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t .
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.62.1 Detailed Description

template<typename T> struct mln::accu::stat::max< T >

Generic max accumulator class. The parameter T is the type of values.

Definition at line 101 of file accu/stat/max.hh.

10.62.2 Member Function Documentation

10.62.2.1 template<typename T > void mln::accu::stat::max< T >::init () [inline]

Manipulators.

Definition at line 146 of file accu/stat/max.hh.

10.62.2.2 template<typename T > bool mln::accu::stat::max< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 196 of file accu/stat/max.hh.

10.62.2.3 template<typename T > void mln::accu::stat::max< T >::set_value (const T & t) [inline]

Force the value of the min to *t*.

Definition at line 180 of file accu/stat/max.hh.

10.62.2.4 void mln::Accumulator< max< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.62.2.5 void mln::Accumulator< max< T > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.62.2.6 `template<typename T> const T & mln::accu::stat::max< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 188 of file `accu/stat/max.hh`.

10.63 `mln::accu::stat::max_h< V > Struct Template Reference`

Generic max function based on histogram over a value set with type `V`.

```
#include <max_h.hh>
```

Inherits `base< const V &, max_h< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const argument & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.63.1 Detailed Description

```
template<typename V> struct mln::accu::stat::max_h< V >
```

Generic max function based on histogram over a value set with type `V`.

Definition at line 100 of file `max_h.hh`.

10.63.2 Member Function Documentation

10.63.2.1 `template<typename V> void mln::accu::stat::max_h< V >::init () [inline]`

Manipulators.

Definition at line 280 of file `max_h.hh`.

10.63.2.2 `template<typename V> bool mln::accu::stat::max_h< V >::is_valid () const` `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 323 of file max_h.hh.

10.63.2.3 `void mln::Accumulator< max_h< V > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.63.2.4 `void mln::Accumulator< max_h< V > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.63.2.5 `template<typename V> const max_h< V >::argument & mln::accu::stat::max_h< V >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 304 of file max_h.hh.

10.64 `mln::accu::stat::mean< T, S, M >` Struct Template Reference

Generic mean accumulator class.

```
#include <mean.hh>
```

Inherits `base< M, mean< T, S, M > >`.

Public Member Functions

- `accu::math::count< T >::result count () const`
Get the cardinality.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `accu::math::sum< T >::result sum () const`
Get the sum of values.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`

Take n times the value t .

- M [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

10.64.1 Detailed Description

**template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>
struct mln::accu::stat::mean< T, S, M >**

Generic mean accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the sum of values; the default type of S is the summation type (property) of T. Parameter M is the type of the mean value; the default type of M is S.

Definition at line 119 of file accu/stat/mean.hh.

10.64.2 Member Function Documentation

**10.64.2.1 template<typename T , typename S , typename M > mln::math::count< T >::result
mln::accu::stat::mean< T, S, M >::count () const [inline]**

Get the cardinality.

Definition at line 242 of file accu/stat/mean.hh.

**10.64.2.2 template<typename T , typename S , typename M > void mln::accu::stat::mean< T, S,
M >::init () [inline]**

Manipulators.

Definition at line 173 of file accu/stat/mean.hh.

**10.64.2.3 template<typename T , typename S , typename M > bool mln::accu::stat::mean< T, S,
M >::is_valid () const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 234 of file accu/stat/mean.hh.

**10.64.2.4 template<typename T , typename S , typename M > mln::math::sum< T >::result
mln::accu::stat::mean< T, S, M >::sum () const [inline]**

Get the sum of values.

Definition at line 251 of file accu/stat/mean.hh.

10.64.2.5 `void mln::Accumulator< mean< T, S, M > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.64.2.6 `void mln::Accumulator< mean< T, S, M > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.64.2.7 `template<typename T , typename S , typename M > M mln::accu::stat::mean< T, S, M >::to_result () const` **[inline]**

Get the value of the accumulator.

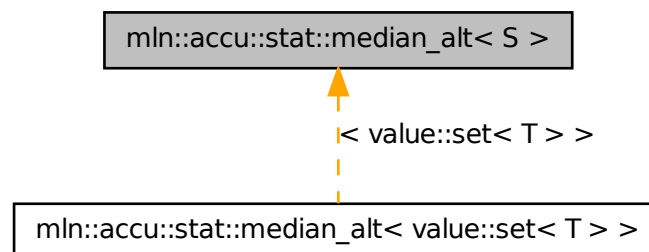
Definition at line 216 of file `accu/stat/mean.hh`.

10.65 `mln::accu::stat::median_alt< S >` Struct Template Reference

Generic `median_alt` function based on histogram over a value set with type S .

`#include <median_alt.hh>`

Inheritance diagram for `mln::accu::stat::median_alt< S >`:



Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`

Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t .

- const argument & [to_result](#) () const

Get the value of the accumulator.

- void [take](#) (const argument &t)

Manipulators.

10.65.1 Detailed Description

template<typename S> struct mln::accu::stat::median_alt< S >

Generic [median_alt](#) function based on histogram over a value set with type S.

Definition at line 54 of file median_alt.hh.

10.65.2 Member Function Documentation

10.65.2.1 template<typename S > bool mln::accu::stat::median_alt< S >::is_valid () const
[inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 282 of file median_alt.hh.

10.65.2.2 template<typename S > void mln::accu::stat::median_alt< S >::take (const argument & t) [inline]

Manipulators.

Definition at line 165 of file median_alt.hh.

10.65.2.3 void mln::Accumulator< median_alt< S > >::take_as_init (const T & t)
[inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.65.2.4 void mln::Accumulator< median_alt< S > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.65.2.5 `template<typename S > const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result () const [inline]`

Get the value of the accumulator.

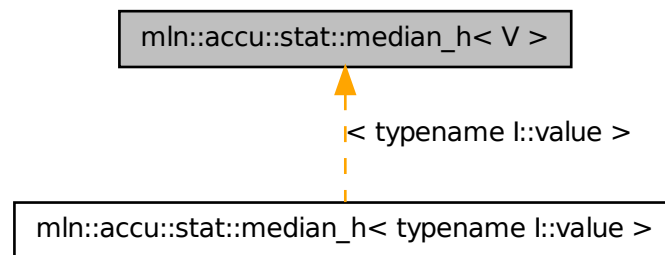
Definition at line 274 of file median_alt.hh.

10.66 mln::accu::stat::median_h< V > Struct Template Reference

Generic median function based on histogram over a value set with type V.

```
#include <median_h.hh>
```

Inheritance diagram for mln::accu::stat::median_h< V >:



Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const argument & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.66.1 Detailed Description

template<typename V> struct mln::accu::stat::median_h< V >

Generic median function based on histogram over a value set with type V.

Definition at line 82 of file median_h.hh.

10.66.2 Member Function Documentation

10.66.2.1 template<typename V> void mln::accu::stat::median_h< V >::init () [inline]

Manipulators.

Definition at line 267 of file median_h.hh.

10.66.2.2 template<typename V> bool mln::accu::stat::median_h< V >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 299 of file median_h.hh.

10.66.2.3 void mln::Accumulator< median_h< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.66.2.4 void mln::Accumulator< median_h< V > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.66.2.5 template<typename V> const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result () const [inline]

Get the value of the accumulator.

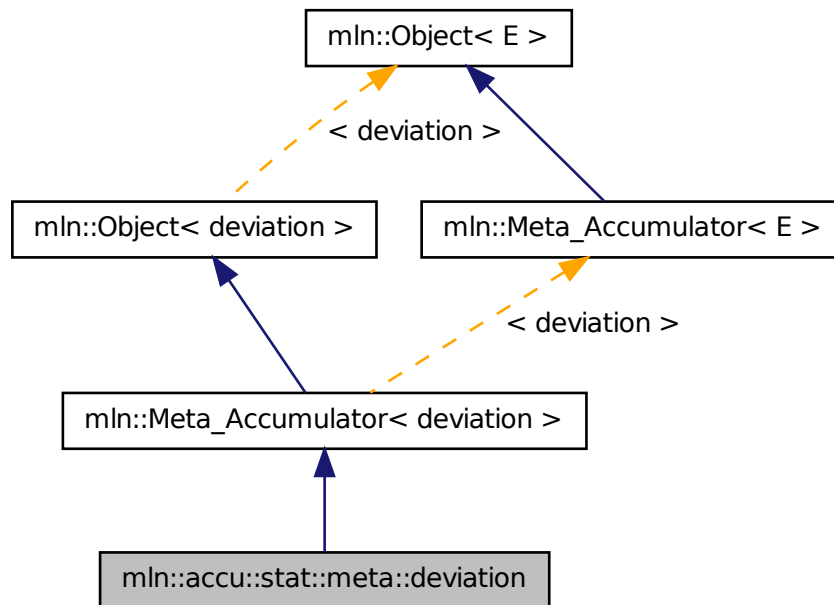
Definition at line 281 of file median_h.hh.

10.67 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

```
#include <deviation.hh>
```

Inheritance diagram for mln::accu::stat::meta::deviation:



10.67.1 Detailed Description

Meta accumulator for deviation.

Definition at line 105 of file deviation.hh.

10.68 mln::accu::stat::min< T > Struct Template Reference

Generic min accumulator class.

```
#include <min.hh>
```

Inherits base< const T &, min< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t.

- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.68.1 Detailed Description

template<typename T> struct mln::accu::stat::min< T >

Generic min accumulator class. The parameter T is the type of values.

Definition at line 102 of file accu/stat/min.hh.

10.68.2 Member Function Documentation

10.68.2.1 template<typename T > void mln::accu::stat::min< T >::init () [inline]

Manipulators.

Definition at line 147 of file accu/stat/min.hh.

**10.68.2.2 template<typename T > bool mln::accu::stat::min< T >::is_valid () const
[inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 195 of file accu/stat/min.hh.

**10.68.2.3 template<typename T > void mln::accu::stat::min< T >::set_value (const T & t)
[inline]**

Force the value of the min to t .

Definition at line 179 of file accu/stat/min.hh.

10.68.2.4 void mln::Accumulator< min< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.68.2.5 `void mln::Accumulator< min< T > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.68.2.6 `template<typename T > const T & mln::accu::stat::min< T >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 187 of file `accu/stat/min.hh`.

10.69 mln::accu::stat::min_h< V > Struct Template Reference

Generic min function based on histogram over a value set with type V .

`#include <min_h.hh>`

Inherits `base< const V &, min_h< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const argument & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.69.1 Detailed Description

`template<typename V> struct mln::accu::stat::min_h< V >`

Generic min function based on histogram over a value set with type V .

Definition at line 100 of file `min_h.hh`.

10.69.2 Member Function Documentation

10.69.2.1 `template<typename V> void mln::accu::stat::min_h< V >::init () [inline]`

Manipulators.

Definition at line 256 of file min_h.hh.

10.69.2.2 `template<typename V> bool mln::accu::stat::min_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 298 of file min_h.hh.

10.69.2.3 `void mln::Accumulator< min_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.69.2.4 `void mln::Accumulator< min_h< V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.69.2.5 `template<typename V> const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result () const [inline]`

Get the value of the accumulator.

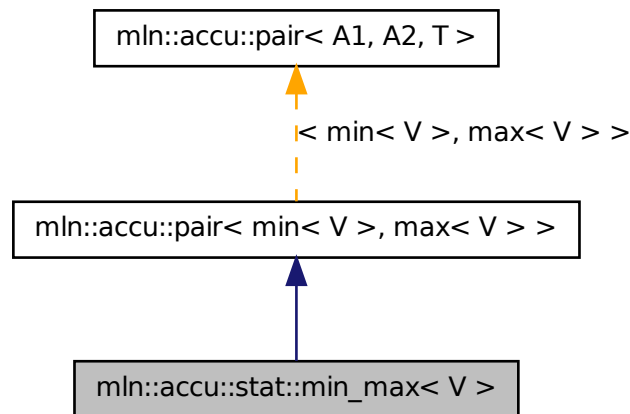
Definition at line 280 of file min_h.hh.

10.70 mln::accu::stat::min_max< V > Struct Template Reference

Generic min and max accumulator class.

```
#include <min_max.hh>
```

Inheritance diagram for `mln::accu::stat::min_max< V >`:



Public Member Functions

- `min< V >::result first () const`
Return the result of the first accumulator.
- `min< V > first_accu () const`
Return the first accumulator.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `max< V >::result second () const`
Return the result of the second accumulator.
- `max< V > second_accu () const`
Return the second accumulator.
- `template<typename T >`
`void take_as_init (const T &t)`
Take as initialization the value t.
- `template<typename T >`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `void init ()`
Manipulators.

- `std::pair< typename min< V >::result, typename max< V >::result > to_result () const`
Get the value of the accumulator.

10.70.1 Detailed Description

template<typename V> struct mln::accu::stat::min_max< V >

Generic min and max accumulator class. The parameter V is the type of values.

Definition at line 61 of file `accu/stat/min_max.hh`.

10.70.2 Member Function Documentation

10.70.2.1 min< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::first () const [inherited]

Return the result of the first accumulator.

10.70.2.2 min< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::first_accu () const [inherited]

Return the first accumulator.

10.70.2.3 void mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::init () [inherited]

Manipulators.

10.70.2.4 bool mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::is_valid () const [inherited]

Check whether this accu is able to return a result.

Always true here.

10.70.2.5 max< V >::result mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::second () const [inherited]

Return the result of the second accumulator.

10.70.2.6 max< V > mln::accu::pair< min< V >, max< V >, mln_argument(min< V >) >::second_accu () const [inherited]

Return the second accumulator.

10.70.2.7 `template<typename E> template<typename T> void mln::Accumulator< E>::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.70.2.8 `template<typename E> template<typename T> void mln::Accumulator< E>::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.70.2.9 `std::pair<typename min< V> ::result, typename max< V> ::result> mln::accu::pair< min< V> , max< V> , mln_argument(min< V>)> ::to_result () const [inherited]`

Get the value of the accumulator.

10.71 mln::accu::stat::rank< T> Struct Template Reference

Generic rank accumulator class.

`#include <rank.hh>`

Inherits `base< const T &, rank< T> >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `unsigned k () const`
Give the rank.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const T & to_result () const`
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.71.1 Detailed Description

template<typename T> struct mln::accu::stat::rank< T >

Generic rank accumulator class. The parameter T is the type of values.

Definition at line 60 of file rank.hh.

10.71.2 Member Function Documentation

10.71.2.1 template<typename T> void mln::accu::stat::rank< T >::init () [inline]

Manipulators.

Definition at line 319 of file rank.hh.

**10.71.2.2 template<typename T> bool mln::accu::stat::rank< T >::is_valid () const
 [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 343 of file rank.hh.

**10.71.2.3 template<typename T> unsigned mln::accu::stat::rank< T >::k () const
 [inline]**

Give the rank.

Definition at line 178 of file rank.hh.

10.71.2.4 void mln::Accumulator< rank< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

**10.71.2.5 void mln::Accumulator< rank< T > >::take_n_times (unsigned n, const T & t)
 [inherited]**

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.71.2.6 `template<typename T> const T & mln::accu::stat::rank< T >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 333 of file rank.hh.

10.72 `mln::accu::stat::rank< bool >` Struct Template Reference

rank accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits base< bool, rank< bool > >.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `bool to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.72.1 Detailed Description

`template<> struct mln::accu::stat::rank< bool >`

rank accumulator class for Boolean.

Definition at line 58 of file rank_bool.hh.

10.72.2 Member Function Documentation

10.72.2.1 `void mln::accu::stat::rank< bool >::init () [inline]`

Manipulators.

Definition at line 105 of file rank_bool.hh.

10.72.2.2 bool mln::accu::stat::rank< bool >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 157 of file rank_bool.hh.

10.72.2.3 void mln::Accumulator< rank< bool > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.72.2.4 void mln::Accumulator< rank< bool > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.72.2.5 bool mln::accu::stat::rank< bool >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 150 of file rank_bool.hh.

10.73 mln::accu::stat::rank_high_quant< T > Struct Template Reference

Generic rank accumulator class.

```
#include <rank_high_quant.hh>
```

Inherits base< const T &, rank_high_quant< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.73.1 Detailed Description

template<typename T> struct mln::accu::stat::rank_high_quant< T >

Generic rank accumulator class. The parameter T is the type of values.

Definition at line 57 of file rank_high_quant.hh.

10.73.2 Member Function Documentation

10.73.2.1 template<typename T > void mln::accu::stat::rank_high_quant< T >::init ()
[inline]

Manipulators.

Definition at line 148 of file rank_high_quant.hh.

10.73.2.2 template<typename T > bool mln::accu::stat::rank_high_quant< T >::is_valid ()
const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file rank_high_quant.hh.

10.73.2.3 void mln::Accumulator< rank_high_quant< T > >::take_as_init (const T & t)
[inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.2.4 void mln::Accumulator< rank_high_quant< T > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.2.5 template<typename T > const T & mln::accu::stat::rank_high_quant< T >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 183 of file rank_high_quant.hh.

10.74 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

```
#include <var.hh>
```

Inherits base< algebra::mat< T::dim, T::dim, float >, var< T > >.

Public Types

- typedef algebra::vec< dim, float > [mean_t](#)

Type equipment.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu returns a valid result.
- [mean_t](#) [mean](#) () const
Get the mean vector.
- unsigned [n_items](#) () const
Get the number of items.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- result [to_result](#) () const
Get the accumulator result (the var value).
- result [variance](#) () const
Get the variance matrix.
- void [init](#) ()
Manipulators.

10.74.1 Detailed Description

```
template<typename T> struct mln::accu::stat::var< T >
```

Var accumulator class. Parameter T is the type of vectors

Definition at line 58 of file accu/stat/var.hh.

10.74.2 Member Typedef Documentation

10.74.2.1 `template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t`

Type equipment.

Definition at line 88 of file `accu/stat/var.hh`.

10.74.3 Member Function Documentation

10.74.3.1 `template<typename T> void mln::accu::stat::var< T >::init () [inline]`

Manipulators.

Definition at line 118 of file `accu/stat/var.hh`.

10.74.3.2 `template<typename T> bool mln::accu::stat::var< T >::is_valid () const [inline]`

Check whether this accu returns a valid result.

Definition at line 213 of file `accu/stat/var.hh`.

10.74.3.3 `template<typename T> var< T >::mean_t mln::accu::stat::var< T >::mean () const [inline]`

Get the mean vector.

Definition at line 200 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

10.74.3.4 `template<typename T> unsigned mln::accu::stat::var< T >::n_items () const [inline]`

Get the number of items.

Definition at line 192 of file `accu/stat/var.hh`.

10.74.3.5 `void mln::Accumulator< var< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.74.3.6 `void mln::Accumulator< var< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.74.3.7 `template<typename T> var< T >::result mln::accu::stat::var< T >::to_result ()`
`const [inline]`

Get the accumulator result (the var value).

Definition at line 168 of file accu/stat/var.hh.

References mln::literal::zero.

10.74.3.8 `template<typename T> var< T >::result mln::accu::stat::var< T >::variance ()`
`const [inline]`

Get the variance matrix.

Definition at line 184 of file accu/stat/var.hh.

10.75 mln::accu::stat::variance< T, S, R > Struct Template Reference

Variance accumulator class.

`#include <variance.hh>`

Inherits base< R, variance< T, S, R > >.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `R mean () const`
Get the mean value.
- `unsigned n_items () const`
Get the number of items.
- `R standard_deviation () const`
Get the standard deviation value.
- `S sum () const`
Get the sum value.
- `void take_as_init (const T &t)`
Take as initialization the value t.
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `R to_result () const`
Get the accumulator result (the variance value).

- `R var () const`
Get the variance value.
- `void init ()`
Manipulators.

10.75.1 Detailed Description

template<typename T, typename S = typename mln::value::props< T >::sum, typename R = S>
struct mln::accu::stat::variance< T, S, R >

Variance accumulator class. Parameter T is the type of values that we sum. Parameter S is the type to store the value sum and the sum of value * value; the default type of S is the summation type (property) of T. Parameter R is the type of the mean and variance values; the default type of R is S.

Definition at line 61 of file variance.hh.

10.75.2 Member Function Documentation

10.75.2.1 template<typename T, typename S, typename R > void mln::accu::stat::variance< T, S, R >::init () [inline]

Manipulators.

Definition at line 125 of file variance.hh.

References mln::literal::zero.

10.75.2.2 template<typename T, typename S, typename R > bool mln::accu::stat::variance< T, S, R >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 229 of file variance.hh.

10.75.2.3 template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::mean () const [inline]

Get the mean value.

Definition at line 187 of file variance.hh.

References mln::literal::zero.

10.75.2.4 template<typename T, typename S, typename R > unsigned mln::accu::stat::variance< T, S, R >::n_items () const [inline]

Get the number of items.

Definition at line 205 of file variance.hh.

10.75.2.5 `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::standard_deviation () const [inline]`

Get the standard deviation value.

Definition at line 221 of file variance.hh.

References mln::accu::stat::variance< T, S, R >::to_result().

10.75.2.6 `template<typename T , typename S , typename R > S mln::accu::stat::variance< T, S, R >::sum () const [inline]`

Get the sum value.

Definition at line 197 of file variance.hh.

10.75.2.7 `void mln::Accumulator< variance< T, S, R > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.75.2.8 `void mln::Accumulator< variance< T, S, R > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.75.2.9 `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::to_result () const [inline]`

Get the accumulator result (the variance value).

Definition at line 176 of file variance.hh.

Referenced by mln::accu::stat::variance< T, S, R >::standard_deviation(), and mln::accu::stat::variance< T, S, R >::var().

10.75.2.10 `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::var () const [inline]`

Get the variance value.

Definition at line 213 of file variance.hh.

References mln::accu::stat::variance< T, S, R >::to_result().

10.76 `mln::accu::tuple< A, n, > Struct Template Reference`

Generic tuple of accumulators.

```
#include <tuple.hh>
```

Inherits base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!)>, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- res [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.76.1 Detailed Description

template<typename A, unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T, boost::tuples::null_type)> struct mln::accu::tuple< A, n, >

Generic tuple of accumulators. The parameter T is the type of values.

Definition at line 74 of file tuple.hh.

10.76.2 Member Function Documentation

**10.76.2.1 template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) >
 void mln::accu::tuple< A, n, >::init () [inline]**

Manipulators.

Definition at line 197 of file tuple.hh.

**10.76.2.2 template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) >
 bool mln::accu::tuple< A, n, >::is_valid () const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 239 of file tuple.hh.

10.76.2.3 `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)>
>::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.76.2.4 `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)>
>::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.76.2.5 `template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) >
tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T) >::res mln::accu::tuple< A, n,
>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 229 of file tuple.hh.

10.77 mln::accu::val< A > Struct Template Reference

Generic val of accumulators.

`#include <v.hh>`

Inherits `base< const A::result &, val< A > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const A::result & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.77.1 Detailed Description

template<typename A> struct mln::accu::val< A >

Generic val of accumulators.

Definition at line 50 of file v.hh.

10.77.2 Member Function Documentation

10.77.2.1 template<typename A > void mln::accu::val< A >::init () [inline]

Manipulators.

Definition at line 119 of file v.hh.

10.77.2.2 template<typename A > bool mln::accu::val< A >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file v.hh.

10.77.2.3 void mln::Accumulator< val< A > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.77.2.4 void mln::Accumulator< val< A > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.77.2.5 template<typename A > const A::result & mln::accu::val< A >::to_result () const [inline]

Get the value of the accumulator.

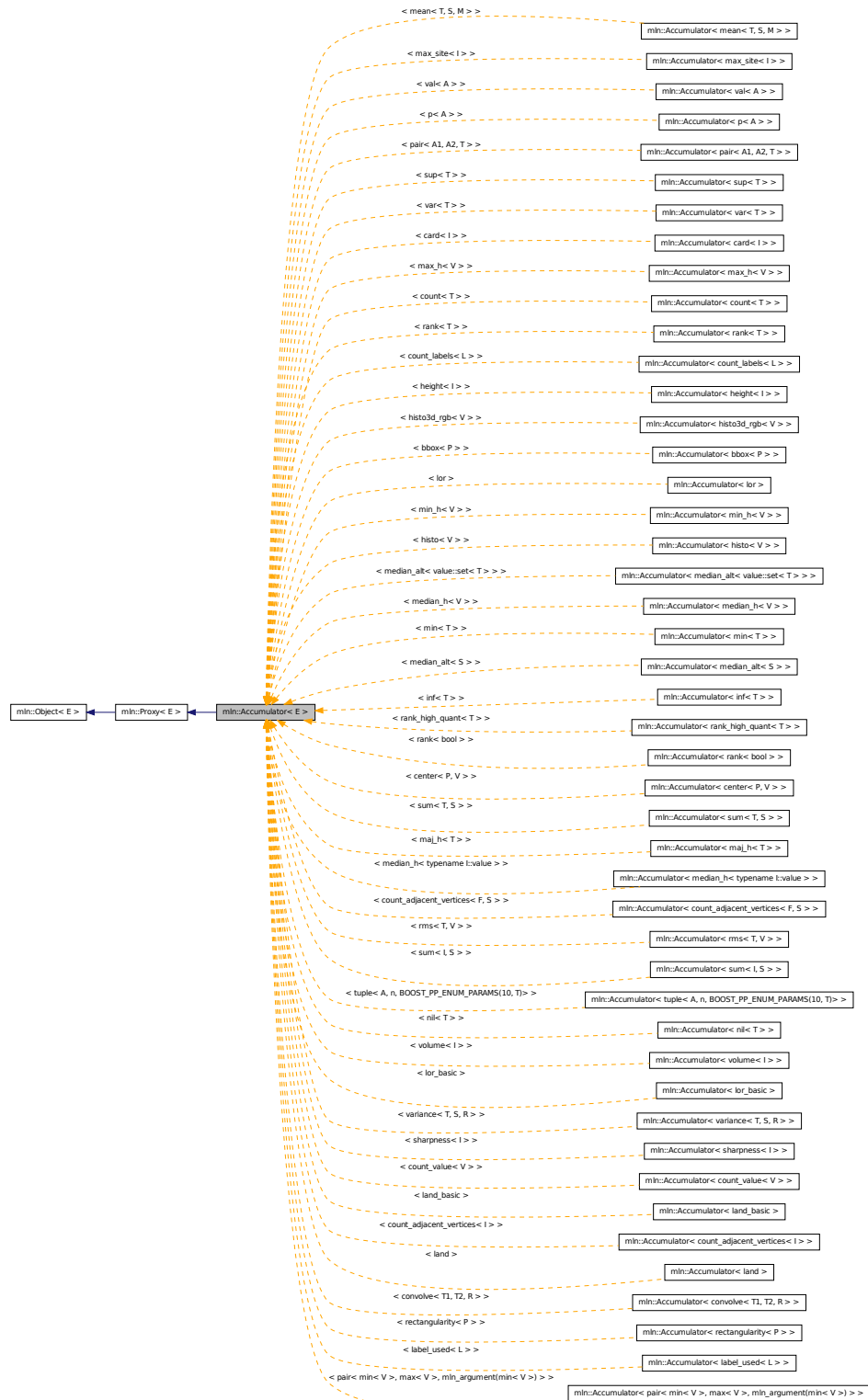
Definition at line 169 of file v.hh.

10.78 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

Inheritance diagram for mln::Accumulator< E >:



Public Member Functions

- `template<typename T >`
`void take_as_init (const T &t)`
Take as initialization the value t .
- `template<typename T >`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t .

10.78.1 Detailed Description

`template<typename E> struct mln::Accumulator< E >`

Base class for implementation of accumulators. The parameter E is the exact type.

See also

[mln::doc::Accumulator](#) for a complete documentation of this class contents.

Definition at line 78 of file accumulator.hh.

10.78.2 Member Function Documentation

10.78.2.1 `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init (const T & t)`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file accumulator.hh.

References `mln::mln_exact()`.

10.78.2.2 `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file accumulator.hh.

References `mln::mln_exact()`.

10.79 mln::algebra::h_mat< d, T > Struct Template Reference

N-Dimensional matrix with homogeneous coordinates.

`#include <h_mat.hh>`

Inherits `mln::algebra::mat< d+1, d+1, T >`.

Public Types

- enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Public Member Functions

- `mat< n, m, T > _1 () const`
Return the inverse of the matrix.
- `h_mat ()`
Constructor without argument.
- `h_mat (const mat< d+1, d+1, T > &x)`
Constructor with the underlying matrix.
- `mat< m, n, T > t () const`
Return the transpose of the matrix.

10.79.1 Detailed Description

template<unsigned d, typename T> struct mln::algebra::h_mat< d, T >

N-Dimensional matrix with homogeneous coordinates.

Definition at line 49 of file algebra/h_mat.hh.

10.79.2 Member Enumeration Documentation

10.79.2.1 template<unsigned d, typename T> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 52 of file algebra/h_mat.hh.

10.79.3 Constructor & Destructor Documentation

10.79.3.1 template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat () [inline]

Constructor without argument.

Definition at line 67 of file algebra/h_mat.hh.

10.79.3.2 template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat (const mat< d+1, d+1, T > & x) [inline]

Constructor with the underlying matrix.

Definition at line 74 of file algebra/h_mat.hh.

10.79.4 Member Function Documentation

10.79.4.1 `template<unsigned n, unsigned m, typename T > mat< n, m, T > mln::algebra::mat< n, m, T >::_1 () const [inline, inherited]`

Return the inverse of the matrix.

Only compile on square matrix.

Definition at line 604 of file algebra/mat.hh.

10.79.4.2 `template<unsigned n, unsigned m, typename T > mat< m, n, T > mln::algebra::mat< n, m, T >::t () const [inline, inherited]`

Return the transpose of the matrix.

Definition at line 538 of file algebra/mat.hh.

10.80 mln::algebra::h_vec< d, C > Class Template Reference

N-Dimensional vector with homogeneous coordinates.

```
#include <h_vec.hh>
```

Inherits mln::algebra::vec< d+1, C >.

Public Types

- enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Public Member Functions

- [h_vec](#) ()
Constructor without argument.
- [h_vec](#) (const vec< d+1, C > &other)
Constructor with the underlying vector.
- template<typename U >
[operator mat< n, 1, U > \(\) const](#)
Conversion to a matrix.
- mat< 1, n, T > [t](#) () const
Transposition.
- vec< d, C > [to_vec](#) () const
Back to the natural (non-homogeneous) space.

Static Public Attributes

- static const vec< n, T > [origin](#) = all_to(0)
Origin value.
- static const vec< n, T > [zero](#) = all_to(0)
Zero value.

10.80.1 Detailed Description

template<unsigned d, typename C> class mln::algebra::h_vec< d, C >

N-Dimensional vector with homogeneous coordinates.

Definition at line 94 of file h_vec.hh.

10.80.2 Member Enumeration Documentation

10.80.2.1 template<unsigned d, typename C> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 98 of file h_vec.hh.

10.80.3 Constructor & Destructor Documentation

10.80.3.1 template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec () [inline]

Constructor without argument.

Definition at line 117 of file h_vec.hh.

References mln::literal::one.

10.80.3.2 template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec (const vec< d+1, C > & other) [inline]

Constructor with the underlying vector.

Definition at line 128 of file h_vec.hh.

10.80.4 Member Function Documentation

10.80.4.1 template<unsigned n, typename T > template<typename U > mln::algebra::vec< n, T >::operator mat< n, 1, U > () const [inline, inherited]

Conversion to a matrix.

Definition at line 368 of file algebra/mat.hh.

10.80.4.2 `template<unsigned n, typename T> mat< 1, n, T> mln::algebra::vec< n, T>::t ()
const [inline, inherited]`

Transposition.

Definition at line 858 of file algebra/mat.hh.

10.80.4.3 `template<unsigned d, typename C> vec< d, C> mln::algebra::h_vec< d, C>::to_vec
() const [inline]`

Back to the natural (non-homogeneous) space.

Definition at line 145 of file h_vec.hh.

10.80.5 Member Data Documentation

10.80.5.1 `template<unsigned n, typename T> const vec< n, T> mln::algebra::vec< n, T>::origin = all_to(0) [static, inherited]`

Origin value.

Definition at line 247 of file algebra/vec.hh.

10.80.5.2 `template<unsigned n, typename T> const vec< n, T> mln::algebra::vec< n, T>::zero
= all_to(0) [static, inherited]`

Zero value.

Definition at line 244 of file algebra/vec.hh.

10.81 mln::bkd_pixter1d< I > Class Template Reference

Backward pixel iterator on a 1-D image with border.

`#include <pixter1d.hh>`

Inherits `backward_pixel_iterator_base_< I, bkd_pixter1d< I > >`.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [bkd_pixter1d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.81.1 Detailed Description

template<typename I> class mln::bkd_pixter1d< I >

Backward pixel iterator on a 1-D image with border.

Definition at line 69 of file pixter1d.hh.

10.81.2 Member Typedef Documentation

10.81.2.1 template<typename I > typedef I mln::bkd_pixter1d< I >::image

[Image](#) type.

Definition at line 76 of file pixter1d.hh.

10.81.3 Constructor & Destructor Documentation

**10.81.3.1 template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d (I & *image*)
[inline]**

Constructor.

Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 117 of file pixter1d.hh.

10.81.4 Member Function Documentation

10.81.4.1 void mln::Iterator< bkd_pixter1d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.82 mln::bkd_pixter2d< I > Class Template Reference

Backward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits backward_pixel_iterator_base_< I, bkd_pixter2d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [bkd_pixter2d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.82.1 Detailed Description

`template<typename I> class mln::bkd_pixter2d< I >`

Backward pixel iterator on a 2-D image with border.

Definition at line 87 of file `pixter2d.hh`.

10.82.2 Member Typedef Documentation

10.82.2.1 `template<typename I > typedef I mln::bkd_pixter2d< I >::image`

[Image](#) type.

Definition at line 94 of file `pixter2d.hh`.

10.82.3 Constructor & Destructor Documentation

10.82.3.1 `template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 169 of file `pixter2d.hh`.

10.82.4 Member Function Documentation

10.82.4.1 `void mln::Iterator< bkd_pixter2d< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.83 mln::bkd_pixter3d< I > Class Template Reference

Backward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits backward_pixel_iterator_base_< I, bkd_pixter3d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [bkd_pixter3d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.83.1 Detailed Description

```
template<typename I> class mln::bkd_pixter3d< I >
```

Backward pixel iterator on a 3-D image with border.

Definition at line 100 of file pixter3d.hh.

10.83.2 Member Typedef Documentation

10.83.2.1 template<typename I > typedef I mln::bkd_pixter3d< I >::image

[Image](#) type.

Definition at line 107 of file pixter3d.hh.

10.83.3 Constructor & Destructor Documentation

10.83.3.1 `template<typename I> mln::bkd_pixter3d< I >::bkd_pixter3d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 207 of file `pixter3d.hh`.

10.83.4 Member Function Documentation

10.83.4.1 `void mln::Iterator< bkd_pixter3d< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

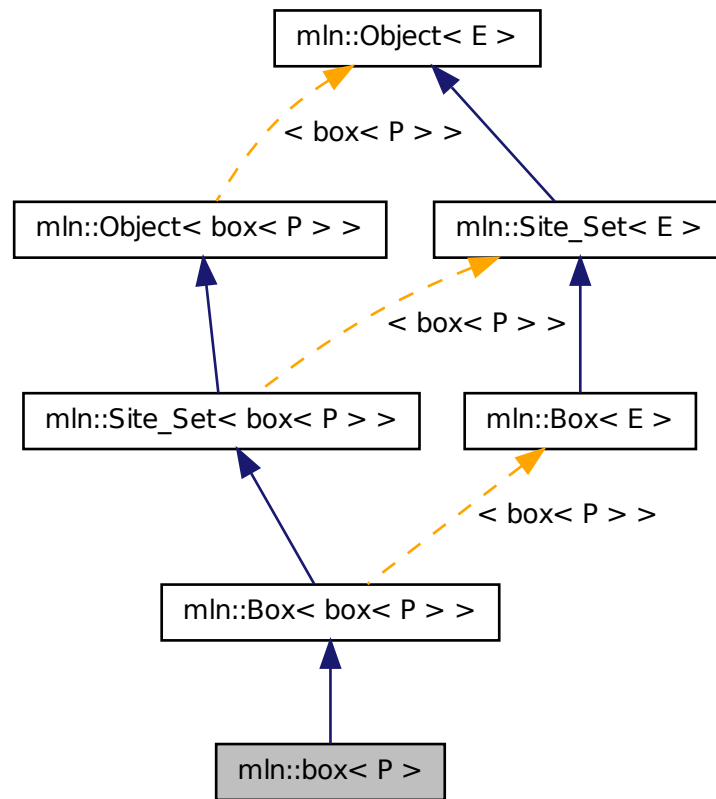
The iterator is valid.

10.84 mln::box< P > Class Template Reference

Generic box class: site set containing points of a regular grid.

```
#include <box.hh>
```

Inheritance diagram for mln::box< P >:



Public Types

- enum
Dimension.
- typedef box_bkd_piter_< P > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef P [element](#)
Element associated type.
- typedef box_fwd_piter_< P > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.

- typedef P [psite](#)
Psite associated type.
- typedef P [site](#)
Site associated type.

Public Member Functions

- const [box](#)< P > & [bbox](#) () const
Give the bounding box of this site set.
- [box](#) ()
Constructor without argument.
- [box](#) (const [site](#) &pmin, const [site](#) &pmax)
*Constructor of a box going from *pmin* to *pmax*.*
- void [crop_wrt](#) (const [box](#)< P > &b)
*Crop this bbox in order to fit in the reference box *b*.*
- void [enlarge](#) (unsigned b)
*Enlarge the box with a border *b*.*
- void [enlarge](#) (unsigned dim, unsigned b)
*Enlarge the box with a border *b* for dimension *dim*.*
- bool [has](#) (const P &p) const
*Test if *p* belongs to the box.*
- bool [is_empty](#) () const
Test if this box is empty.
- bool [is_valid](#) () const
*Test that the box owns valid data, i.e., is initialized and with *pmin* being 'less-than' *pmax*.*
- unsigned [len](#) (unsigned i) const
*Give the length of the *i*-th side of the box.*
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- void [merge](#) (const [box](#)< P > &b)
Merge inplace with another box.
- unsigned [nsites](#) () const
Give the number of sites of this box.

- `P pcenter () const`
Return the approximated central site of this box.
- `P & pmax ()`
Reference to the maximum point.
- `P pmax () const`
Maximum point.
- `P pmin () const`
Minimum point.
- `P & pmin ()`
Reference to the minimum point.
- `box< P > to_larger (unsigned b) const`
Give a larger box.
- `box (typename P::coord ninds)`

Related Functions

(Note that these are not member functions.)

- `template<typename P > std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`
*Print a generic box *b* into the output stream *ostr*.*

10.84.1 Detailed Description

`template<typename P> class mln::box< P >`

Generic box class: site set containing points of a regular grid. Parameter `P` is the corresponding type of point.

Definition at line 81 of file `core/site_set/box.hh`.

10.84.2 Member Typedef Documentation

10.84.2.1 `template<typename P> typedef box_bkd_piter_<P> mln::box< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 105 of file `core/site_set/box.hh`.

10.84.2.2 `template<typename P> typedef P mln::box< P >::element`

Element associated type.

Definition at line 90 of file `core/site_set/box.hh`.

10.84.2.3 `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 99 of file core/site_set/box.hh.

10.84.2.4 `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 102 of file core/site_set/box.hh.

10.84.2.5 `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

Definition at line 93 of file core/site_set/box.hh.

10.84.2.6 `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

Definition at line 96 of file core/site_set/box.hh.

10.84.3 Member Enumeration Documentation

10.84.3.1 `template<typename P> anonymous enum`

Dimension.

Definition at line 87 of file core/site_set/box.hh.

10.84.4 Constructor & Destructor Documentation

10.84.4.1 `template<typename P> mln::box< P >::box () [inline]`

Constructor without argument.

Definition at line 275 of file core/site_set/box.hh.

10.84.4.2 `template<typename P> mln::box< P >::box (const site & pmin, const site & pmax) [inline]`

Constructor of a box going from *pmin* to *pmax*.

Definition at line 284 of file core/site_set/box.hh.

References `mln::box< P >::is_valid()`.

10.84.4.3 `template<typename P> mln::box< P >::box (typename P::coord ninds) [inline, explicit]`

Constructors with different numbers of arguments (sizes) w.r.t. the dimension.

Definition at line 293 of file core/site_set/box.hh.

References mln::literal::origin.

10.84.5 Member Function Documentation

10.84.5.1 const box< P > & mln::Box< box< P > >::bbox () const [inherited]

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.84.5.2 template<typename P > void mln::box< P >::crop_wrt (const box< P > & b) [inline]

Crop this bbox in order to fit in the reference box b.

Definition at line 205 of file core/site_set/box.hh.

References mln::box< P >::pmax(), and mln::box< P >::pmin().

Referenced by mln::debug::draw_graph(), and mln::make_debug_graph_image().

10.84.5.3 template<typename P > void mln::box< P >::enlarge (unsigned b) [inline]

Enlarge the box with a border b.

Definition at line 337 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

Referenced by mln::registration::icp().

10.84.5.4 template<typename P > void mln::box< P >::enlarge (unsigned dim, unsigned b) [inline]

Enlarge the box with a border b for dimension dim.

Definition at line 351 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.84.5.5 template<typename P > bool mln::box< P >::has (const P & p) const [inline]

Test if p belongs to the box.

Parameters

[in] *p* A point site.

Definition at line 325 of file core/site_set/box.hh.

References `mln::box< P >::is_valid()`.

Referenced by `mln::morpho::line_gradient()`.

10.84.5.6 `bool mln::Box< box< P > >::is_empty () const [inherited]`

Test if this box is empty.

10.84.5.7 `template<typename P> bool mln::box< P >::is_valid () const [inline]`

Test that the box owns valid data, i.e., is initialized and with `pmin` being 'less-than' `pmax`.

Definition at line 195 of file core/site_set/box.hh.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::enlarge()`, `mln::box< P >::has()`, `mln::box< P >::merge()`, `mln::box< P >::pcenter()`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `mln::box< P >::to_larger()`.

10.84.5.8 `unsigned mln::Box< box< P > >::len (unsigned i) const [inherited]`

Give the length of the `i`-th side of the box.

Precondition

`i < site::dim`

Warning

This method is final for all box classes.

10.84.5.9 `template<typename P> std::size_t mln::box< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 407 of file core/site_set/box.hh.

10.84.5.10 `template<typename P> void mln::box< P >::merge (const box< P > & b) [inline]`

Merge inplace with another box.

Definition at line 221 of file core/site_set/box.hh.

References `mln::box< P >::is_valid()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

10.84.5.11 `unsigned mln::Box< box< P > >::nsites () const [inherited]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.84.5.12 `template<typename P> P mln::box< P>::pcenter () const [inline]`

Return the approximated central site of this box.

Definition at line 395 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.84.5.13 `template<typename P> P mln::box< P >::pmax () const [inline]`

Maximum point.

Definition at line 259 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

Referenced by `mln::box<P>::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, `mln::io::fld::load()`, and `mln::box<P>::merge()`.

10.84.5.14 `template<typename P> P & mln::box< P >::pmax () [inline]`

Reference to the maximum point.

Definition at line 268 of file core/site_set/box.hh.

10.84.5.15 `template<typename P> P & mln::box< P>::pmin () [inline]`

Reference to the minimum point.

Definition at line 251 of file core/site_set/box.hh.

```
10.84.5.16 template<typename P> P mln::box< P >::pmin ( ) const [inline]
```

Minimum point.

Definition at line 242 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

Referenced by `mln::box< P >::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, `mln::io::fld::load()`, and `mln::box< P >::merge()`.

```
10.84.5.17 template<typename P> box< P> mln::box< P>::to_larger ( unsigned b ) const
[inline]
```

Give a larger box.

Definition at line 378 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.84.6 Friends And Related Function Documentation

10.84.6.1 `template<typename P> std::ostream & operator<< (std::ostream & ostr, const box<P> & b)` [**related**]

Print a generic box *b* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *b* A generic box.

Returns

The modified output stream *ostr*.

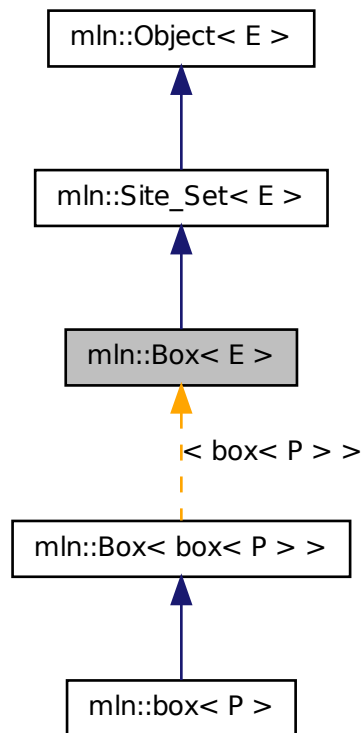
Definition at line 414 of file core/site_set/box.hh.

10.85 mln::Box< E > Struct Template Reference

Base class for implementation classes of boxes.

```
#include <box.hh>
```

Inheritance diagram for mln::Box< E >:



Public Member Functions

- `const E & bbox () const`
Give the bounding box of this site set.
- `bool is_empty () const`
Test if this box is empty.
- `unsigned len (unsigned i) const`
*Give the length of the *i*-th side of the box.*
- `unsigned nsites () const`
Give the number of sites of this box.

Related Functions

(Note that these are not member functions.)

- `template<typename SI, typename Sr >`
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of point sets.
- `template<typename SI, typename Sr >`
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Strict inclusion test between site sets lhs and rhs.
- `template<typename BI, typename Br >`
`bool operator< (const Box< BI > &lhs, const Box< Br > &rhs)`
Strict inclusion test between boxes lhs and rhs.
- `template<typename S >`
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`
Print a site set set into the output stream ostr.
- `template<typename SI, typename Sr >`
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename BI, typename Br >`
`bool operator<= (const Box< BI > &lhs, const Box< Br > &rhs)`
Inclusion test between boxes lhs and rhs.
- `template<typename SI, typename Sr >`
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Union of a couple of point sets.
- `template<typename S >`
`p_set< typename S::site > unique (const Site_Set< S > &s)`
Give the unique set of s.

10.85.1 Detailed Description

template<typename E> struct mln::Box< E >

Base class for implementation classes of boxes. Boxes are particular site sets useful to bound any set of sites defined on a regular grid.

See also

[mln::doc::Box](#) for a complete documentation of this class contents.

Definition at line 48 of file core/concept/box.hh.

10.85.2 Member Function Documentation**10.85.2.1 template<typename E > const E & mln::Box< E >::bbox () const [inline]**

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 124 of file core/concept/box.hh.

10.85.2.2 template<typename E > bool mln::Box< E >::is_empty () const [inline]

Test if this box is empty.

Definition at line 167 of file core/concept/box.hh.

10.85.2.3 template<typename E > unsigned mln::Box< E >::len (unsigned i) const [inline]

Give the length of the *i*-th side of the box.

Precondition

i < site::dim

Warning

This method is final for all box classes.

Definition at line 131 of file core/concept/box.hh.

10.85.2.4 template<typename E > unsigned mln::Box< E >::nsites () const [inline]

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 153 of file core/concept/box.hh.

Referenced by mln::morpho::line_gradient().

10.85.3 Friends And Related Function Documentation

10.85.3.1 `template<typename SI , typename Sr > p_set< typename SI::site > diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

10.85.3.2 `template<typename SI , typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

10.85.3.3 `template<typename SI , typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

10.85.3.4 `template<typename BI , typename Br > bool operator< (const Box< BI > & lhs, const Box< Br > & rhs) [related]`

Strict inclusion test between boxes `lhs` and `rhs`.

Parameters

[in] *lhs* A box (strictly included?).

[in] *rhs* Another box (includer?).

Definition at line 193 of file `core/concept/box.hh`.

10.85.3.5 `template<typename S > std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set) [related, inherited]`

Print a site set `set` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

10.85.3.6 `template<typename Sl, typename Sr > bool operator<= (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [**related**, **inherited**]

Inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

Definition at line 491 of file operators.hh.

10.85.3.7 `template<typename Bl, typename Br > bool operator<= (const Box< Bl > & lhs, const Box< Br > & rhs)` [**related**]

Inclusion test between boxes `lhs` and `rhs`.

Parameters

[in] *lhs* A box (included?).

[in] *rhs* Another box (includer?).

Definition at line 178 of file core/concept/box.hh.

10.85.3.8 `template<typename Sl, typename Sr > bool operator== (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [**related**, **inherited**]

Equality test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set.

[in] *rhs* Another site set.

Definition at line 467 of file operators.hh.

10.85.3.9 `template<typename Sl, typename Sr > p_set< typename Sl::site > sym_diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [**related**, **inherited**]

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.85.3.10 `template<typename Sl, typename Sr > p_set< typename Sl::site > uni (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [**related**, **inherited**]

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.85.3.11 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s) [related, inherited]`

Give the unique set of `s`.

Definition at line 61 of file `unique.hh`.

10.86 `mln::box_runend_piter< P >` Class Template Reference

A generic backward iterator on points by lines.

`#include <box_runend_piter.hh>`

Inherits `site_set_iterator_base< box< P >, box_runend_piter< P > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `unsigned run_length () const`
Give the lenght of the run.

10.86.1 Detailed Description

`template<typename P> class mln::box_runend_piter< P >`

A generic backward iterator on points by lines. The parameter `P` is the type of points.

Definition at line 49 of file `box_runend_piter.hh`.

10.86.2 Member Function Documentation

10.86.2.1 `void mln::Site_Iterator< box_runend_piter< P > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.86.2.2 `template<typename P> unsigned mln::box_runend_piter< P >::run_length () const [inline]`

Give the lenght of the run.

Definition at line 167 of file `box_runend_piter.hh`.

10.87 mln::box_runstart_piter< P > Class Template Reference

A generic forward iterator on points by lines.

```
#include <box_runstart_piter.hh>
```

Inherits site_set_iterator_base< box< P >, box_runstart_piter< P > >.

Public Member Functions

- [box_runstart_piter](#) (const [box](#)< P > &b)

Constructor.

- void [next](#) ()

Go to the next element.

- unsigned [run_length](#) () const

Give the lenght of the run.

10.87.1 Detailed Description

```
template<typename P> class mln::box_runstart_piter< P >
```

A generic forward iterator on points by lines. The parameter P is the type of points.

Definition at line 49 of file box_runstart_piter.hh.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 `template<typename P> mln::box_runstart_piter< P >::box_runstart_piter (const box< P > & b) [inline]`

Constructor.

Parameters

[in] *b* A box.

Definition at line 105 of file box_runstart_piter.hh.

10.87.3 Member Function Documentation

10.87.3.1 `void mln::Site_Iterator< box_runstart_piter< P > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.87.3.2 `template<typename P> unsigned mln::box_runstart_piter< P >::run_length () const [inline]`

Give the length of the run.

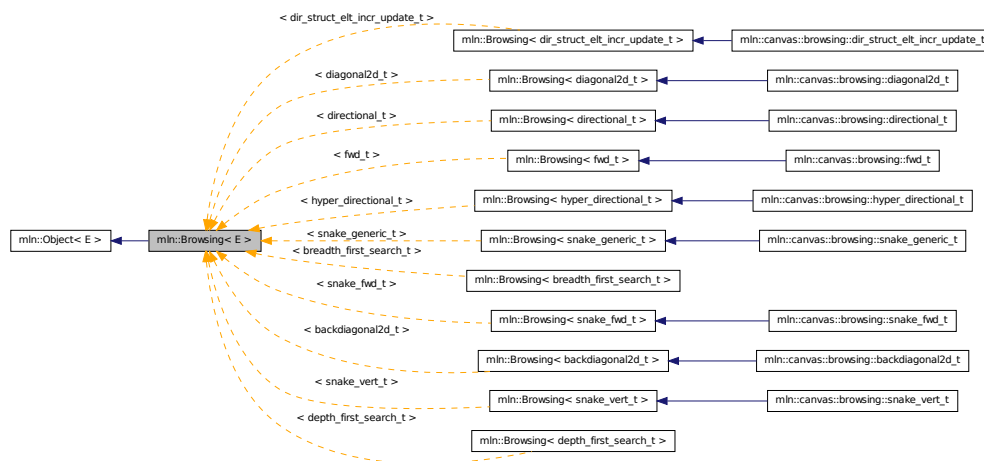
Definition at line 167 of file `box_runstart_piter.hh`.

10.88 `mln::Browsing< E >` Struct Template Reference

Base class for implementation classes that are browsings.

```
#include <browsing.hh>
```

Inheritance diagram for `mln::Browsing< E >`:



10.88.1 Detailed Description

```
template<typename E> struct mln::Browsing< E >
```

Base class for implementation classes that are browsings.

See also

`mln::doc::Browsing` for a complete documentation of this class contents.

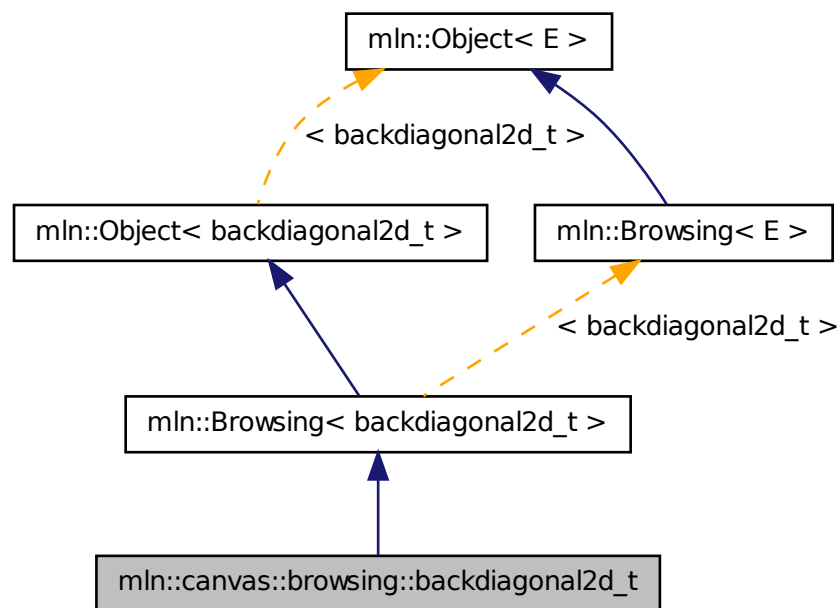
Definition at line 56 of file `browsing.hh`.

10.89 mln::canvas::browsing::backdiagonal2d_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <backdiagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::backdiagonal2d_t:



10.89.1 Detailed Description

[Browsing](#) in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
```

```
--- as types:
```

```
I;
```

```

--- as attributes:
dim;
dir; // and test dir < dim
input;
p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```

-----> | 4 7 9 | 2 5 8 | 1 3 6

```

Definition at line 83 of file backdiagonal2d.hh.

10.90 mln::canvas::browsing::breadth_first_search_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <breadth_first_search.hh>
```

Inherits graph_first_search_t< breadth_first_search_t, std::queue >.

10.90.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

Definition at line 79 of file breadth_first_search.hh.

10.91 mln::canvas::browsing::depth_first_search_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <depth_first_search.hh>
```

Inherits graph_first_search_t< depth_first_search_t, std::stack >.

10.91.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

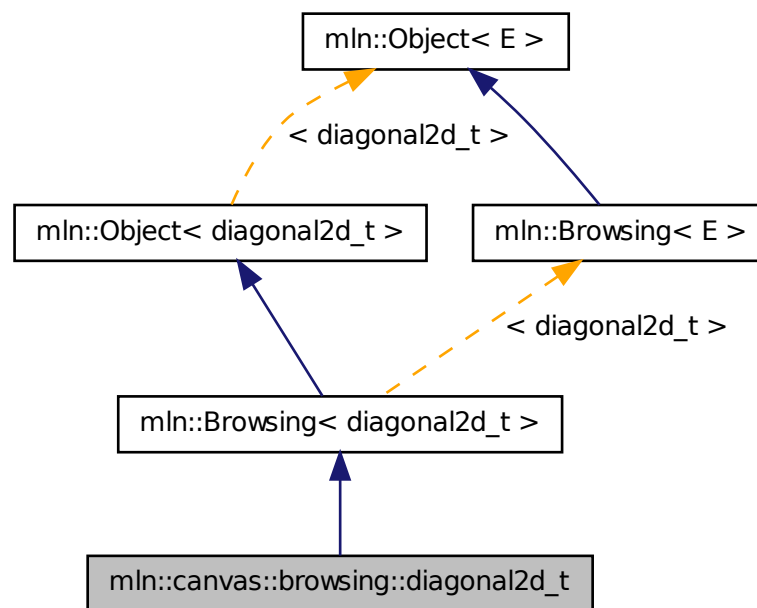
Definition at line 80 of file depth_first_search.hh.

10.92 mln::canvas::browsing::diagonal2d_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <diagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::diagonal2d_t:



10.92.1 Detailed Description

[Browsing](#) in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
```

```
--- as types:
```

```
I;
```

```

--- as attributes:
dim;
dir; // and test dir < dim
input;
p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```
| 1 3 6 | 2 5 8 | 4 7 9 L----->
```

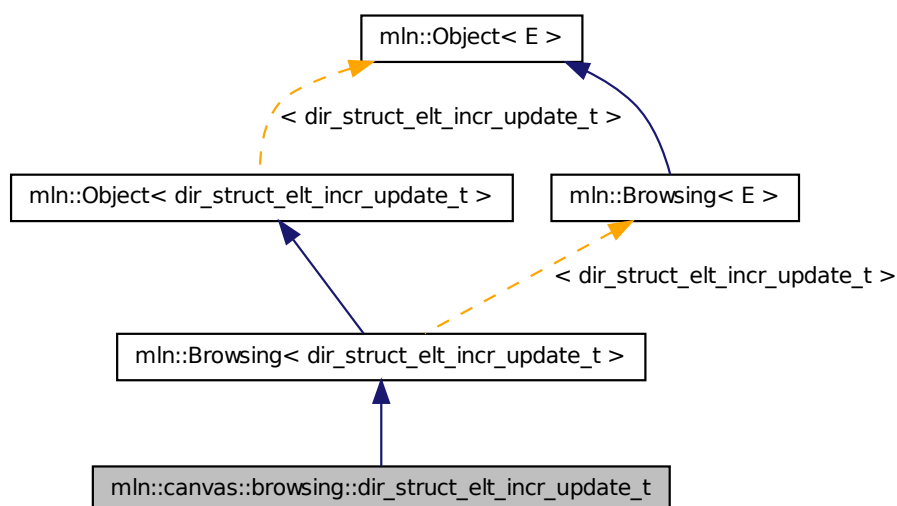
Definition at line 82 of file diagonal2d.hh.

10.93 mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference

[Browsing](#) in a certain direction with a segment.

```
#include <dir_struct_elt_incr_update.hh>
```

Inheritance diagram for mln::canvas::browsing::dir_struct_elt_incr_update_t:



10.93.1 Detailed Description

[Browsing](#) in a certain direction with a segment. This canvas browse all the point of an image 'input' of type 'I', of dimension 'dim' in the direction 'dir' with considering weigh the 'length' nearest points.

The functor should provide (In addition to 'input', 'I', 'dim', 'dir' and 'length') six methods :

- `init()` : Will be called at the beginning.
- `init_line()` : Will be called at the beginning of each line.
- `add_point(q)` : Will be called for taking the new point 'q' into account.
- `remove_point(q)` : Will be called for untaking the new point 'q' into account.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;
p;
length;
--- as methods:
void init();
void init_line();
void add_point(q)
void remove_point(q)
void next();
void final();
}
```

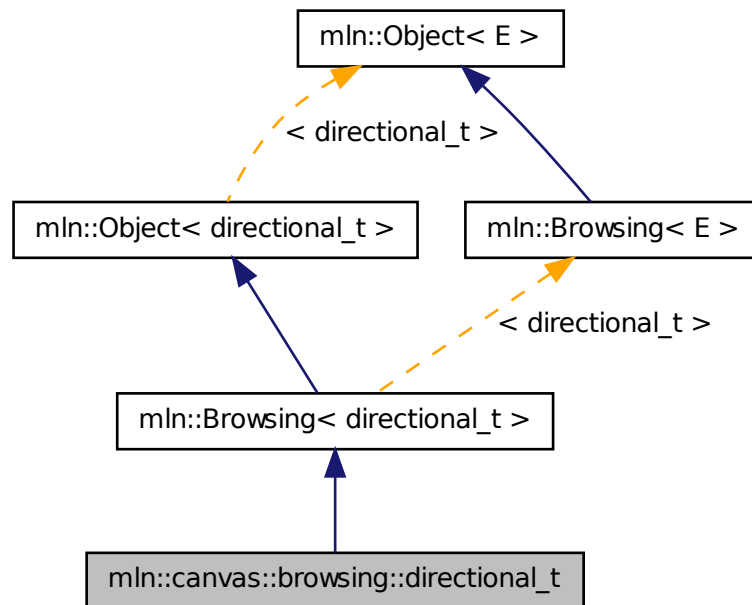
Definition at line 86 of file `dir_struct_elt_incr_update.hh`.

10.94 mln::canvas::browsing::directional_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for mln::canvas::browsing::directional_t:



10.94.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```

1 0 0 2 0 0 3 0 0
4 0 0 5 0 0 6 0 0
7 0 0 8 0 0 9 0 0

```

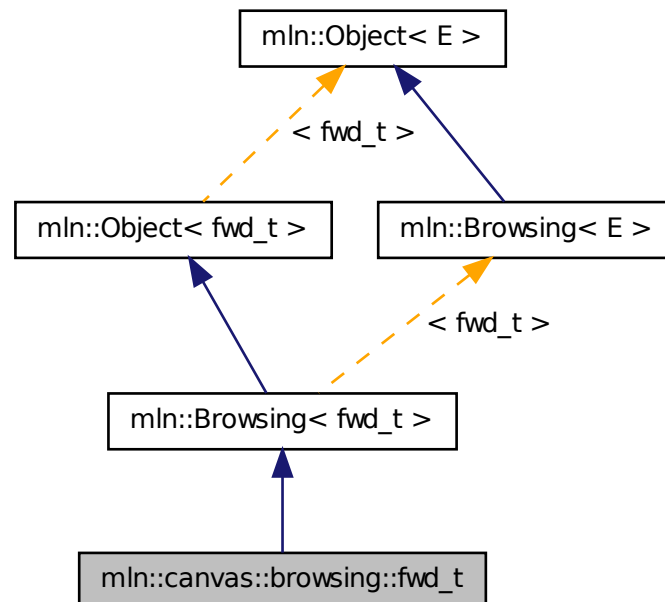
Definition at line 90 of file directional.hh.

10.95 mln::canvas::browsing::fwd_t Struct Reference

Canvas for forward browsing.

```
#include <fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::fwd_t:



10.95.1 Detailed Description

Canvas for forward browsing. This canvas browse all the points of an image 'input' of type 'I' from left to right and from top to bottom

The functor should provide (In addition of 'I' and 'input') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall feature:

```
{  
--- as typedef:  
I;  
--as attributes:  
input;  
p;  
--- as method:  
void init();  
void next();  
void final();  
}
```

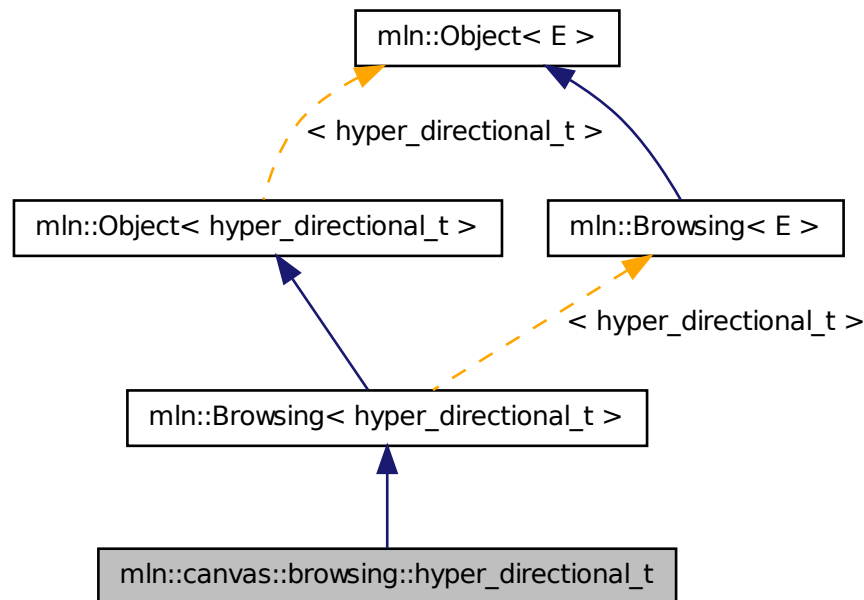
Definition at line 72 of file fwd.hh.

10.96 mln::canvas::browsing::hyper_directional_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <hyper_directional.hh>
```

Inheritance diagram for mln::canvas::browsing::hyper_directional_t:



10.96.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

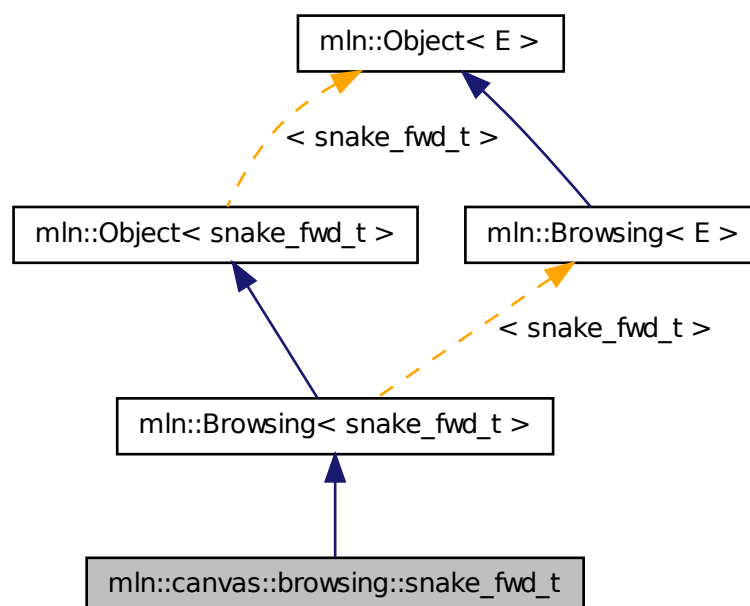
Definition at line 74 of file hyper_directional.hh.

10.97 mln::canvas::browsing::snake_fwd_t Struct Reference

[Browsing](#) in a snake-way, forward.

```
#include <snake_fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_fwd_t:



10.97.1 Detailed Description

[Browsing](#) in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
-----> <-----' '----->
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down. (will also be called once at the first point).
- `fwd()` : Will be called after each moving right.
- `bwd()` : Will be called after each moving left.

This methods should access to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{  
--- as attributes:  
input;  
p;  
--- as methods:  
void init();  
void down();  
void fwd();  
void bkd();  
}
```

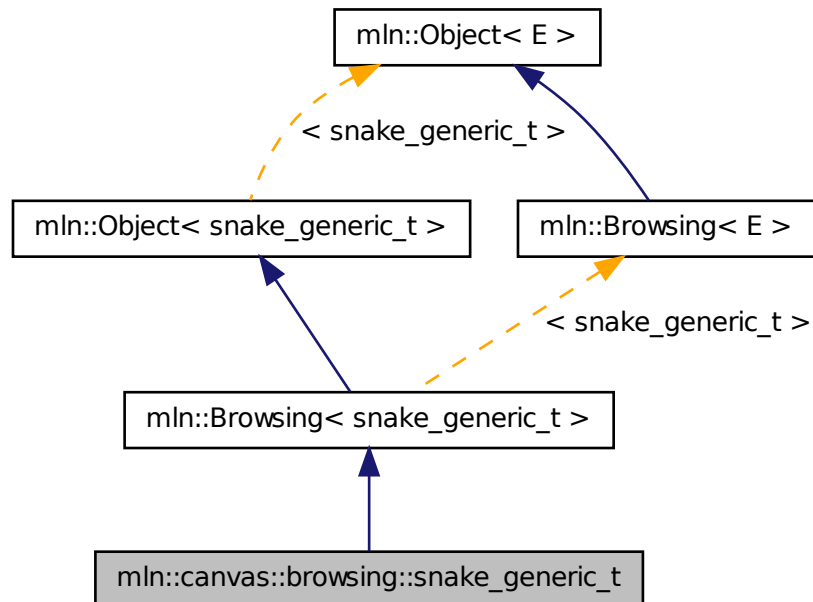
Definition at line 84 of file `snake_fwd.hh`.

10.98 mln::canvas::browsing::snake_generic_t Struct Reference

Multidimensional [Browsing](#) in a given-way.

```
#include <snake_generic.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_generic_t:



10.98.1 Detailed Description

Multidimensional [Browsing](#) in a given-way. F shall feature:

```

{
--- as attributes:
input;
p;
--- as methods:
void init();
void *() moves[];
dpsite dps[];
}
  
```

init is called before browsing

The snake follow dimension using the delta point site of dps. dps[0] = delta psite following the global dimension (forward) dps[1] = delta psite following the 2nd dimension to follow (forward). dps[2] = delta psite following the 2nd dimension to follow (backward). dps[3] = delta psite following the 3rd dimension to follow (forward). dps[3] = delta psite following the 3rd dimension to follow (backward).

moves contains pointer to f's members. These members will be call in each time the snake progress in the

correct dimension :

moves[i] is called at each move following the delta psite dps[i]

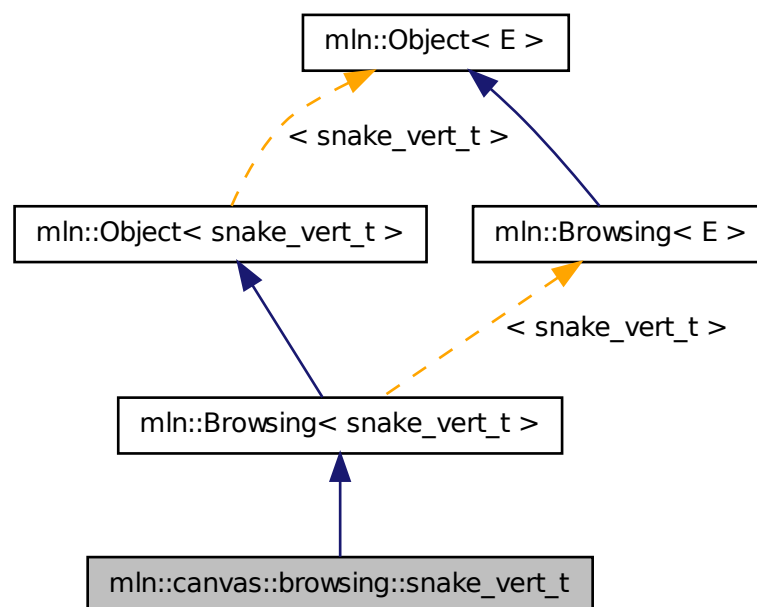
Definition at line 76 of file snake_generic.hh.

10.99 mln::canvas::browsing::snake_vert_t Struct Reference

Browsing in a snake-way, forward.

```
#include <snake_vert.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_vert_t:



10.99.1 Detailed Description

Browsing in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

| A | I | I | V | V |

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down.
- `up()` : Will be called after each moving up.

- fwd() : Will be called after each moving right. (will also be called once at the first point).

This methods should acces to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{
--- as attributes:
input;
p;
--- as methods:
void init();
void down();
void up();
void fwd();
}
```

Definition at line 83 of file snake_vert.hh.

10.100 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

```
#include <chamfer.hh>
```

10.100.1 Detailed Description

```
template<typename F> struct mln::canvas::chamfer< F >
```

Compute chamfer distance.

Definition at line 47 of file canvas/chamfer.hh.

10.101 mln::category< R(*) (A) > Struct Template Reference

Category declaration for a unary C function.

```
#include <c.hh>
```

10.101.1 Detailed Description

```
template<typename R, typename A> struct mln::category< R(*) (A) >
```

Category declaration for a unary C function.

Definition at line 51 of file c.hh.

10.102 mln::complex_image< D, G, V > Class Template Reference

[Image](#) based on a complex.

```
#include <complex_image.hh>
```

Inherits [image_primary](#)< V, [p_complex](#)< D, G >, [complex_image](#)< D, G, V > >.

Public Types

- typedef G [geom](#)
The geometry type of the complex.
- typedef V & [lvalue](#)
Return type of read-write access.
- typedef const V & [rvalue](#)
Return type of read-only access.
- typedef [complex_image](#)< D, tag::psite_< G >, tag::value_< V > > [skeleton](#)
Skeleton.
- typedef V [value](#)
Value associated type.

Public Member Functions

- [rvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p) const
Read-only access of face value at point site p.
- [lvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p)
Read-write access of face value at point site p.
- [complex_image](#) ()
Constructors.
- const [p_complex](#)< D, G > & [domain](#) () const
Accessors.
- const [metal::vec](#)< D+1, [std::vector](#)< [mlc_unbool](#)(V) > > & [values](#) () const
Return the array of values associated to the faces.

Static Public Attributes

- static const unsigned [dim](#) = D
The dimension of the complex.

10.102.1 Detailed Description

template<unsigned D, typename G, typename V> class mln::complex_image< D, G, V >

[Image](#) based on a complex. Values attached to each face of the complex.

Template Parameters

- D* The dimension of the complex.
- G* The geometry type of the complex.
- V* The value type of the image.

Definition at line 164 of file mln/core/image/complex_image.hh.

10.102.2 Member Typedef Documentation

10.102.2.1 template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom

The geometry type of the complex.

Definition at line 172 of file mln/core/image/complex_image.hh.

10.102.2.2 template<unsigned D, typename G, typename V> typedef V& mln::complex_image< D, G, V >::lvalue

Return type of read-write access.

Definition at line 177 of file mln/core/image/complex_image.hh.

10.102.2.3 template<unsigned D, typename G, typename V> typedef const V& mln::complex_image< D, G, V >::rvalue

Return type of read-only access.

Definition at line 180 of file mln/core/image/complex_image.hh.

10.102.2.4 template<unsigned D, typename G, typename V> typedef complex_image< D, tag::psite_<G>, tag::value_<V> > mln::complex_image< D, G, V >::skeleton

Skeleton.

Definition at line 183 of file mln/core/image/complex_image.hh.

10.102.2.5 template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value

[Value](#) associated type.

Definition at line 174 of file mln/core/image/complex_image.hh.

10.102.3 Constructor & Destructor Documentation

10.102.3.1 `template<unsigned D, typename G , typename V > mln::complex_image< D, G, V >::complex_image () [inline]`

Constructors.

Definition at line 276 of file mln/core/image/complex_image.hh.

10.102.4 Member Function Documentation

10.102.4.1 `template<unsigned D, typename G , typename V > const p_complex< D, G > & mln::complex_image< D, G, V >::domain () const [inline]`

Accessors.

Return the domain of psites of the image.

Definition at line 343 of file mln/core/image/complex_image.hh.

10.102.4.2 `template<unsigned D, typename G, typename V > complex_image< D, G, V >::lvalue mln::complex_image< D, G, V >::operator() (const complex_psite< D, G > & p) [inline]`

Read-write access of face value at point site p.

Definition at line 326 of file mln/core/image/complex_image.hh.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

10.102.4.3 `template<unsigned D, typename G, typename V > complex_image< D, G, V >::rvalue mln::complex_image< D, G, V >::operator() (const complex_psite< D, G > & p) const [inline]`

Read-only access of face value at point site p.

Definition at line 317 of file mln/core/image/complex_image.hh.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

10.102.4.4 `template<unsigned D, typename G , typename V > const metal::vec< D+1, std::vector< mlc_unbool(V) > > & mln::complex_image< D, G, V >::values () const [inline]`

Return the array of values associated to the faces.

Definition at line 335 of file mln/core/image/complex_image.hh.

10.102.5 Member Data Documentation

10.102.5.1 `template<unsigned D, typename G, typename V> const unsigned mln::complex_image< D, G, V >::dim = D [static]`

The dimension of the complex.

Definition at line 170 of file mln/core/image/complex_image.hh.

10.103 mln::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference

Backward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits `site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >`.

Public Types

- typedef N::complex_bkd_iter [iter_type](#)
The type of the underlying complex iterator.
- typedef N::psite [psite](#)
The [Pseudo_Site](#) type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_neighborhood_bkd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.103.1 Detailed Description

```
template<typename I, typename G, typename N> class mln::complex_neighborhood_bkd_piter< I, G, N >
```

Backward iterator on complex neighborhood.

Definition at line 124 of file `complex_neighborhood_piter.hh`.

10.103.2 Member Typedef Documentation

10.103.2.1 `template<typename I, typename G, typename N> typedef N::complex_bkd_iter mln::complex_neighborhood_bkd_piter< I, G, N >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file `complex_neighborhood_piter.hh`.

10.103.2.2 `template<typename I, typename G, typename N> typedef N ::psite
mln::complex_neighborhood_bkd_piter< I, G, N >::psite`

The [Pseudo_Site](#) type.

Definition at line 133 of file `complex_neighborhood_piter.hh`.

10.103.3 Constructor & Destructor Documentation

10.103.3.1 `template<typename I , typename G , typename N > mln::complex_
neighborhood_bkd_piter< I, G, N >::complex_neighborhood_bkd_piter ()
[inline]`

Construction.

Definition at line 305 of file `complex_neighborhood_piter.hh`.

10.103.4 Member Function Documentation

10.103.4.1 `template<typename I , typename G , typename N > const N::complex_bkd_iter &
mln::complex_neighborhood_bkd_piter< I, G, N >::iter () const [inline]`

Accessors.

Definition at line 382 of file `complex_neighborhood_piter.hh`.

10.103.4.2 `void mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.104 mln::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference

Forward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits `site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >`.

Public Types

- `typedef N::complex_fwd_iter` [iter_type](#)

The type of the underlying complex iterator.

- typedef N::psite [psite](#)
The [Pseudo_Site](#) type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_neighborhood_fwd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.104.1 Detailed Description

template<typename I, typename G, typename N> class mln::complex_neighborhood_fwd_piter< I, G, N >

Forward iterator on complex neighborhood.

Definition at line 53 of file `complex_neighborhood_piter.hh`.

10.104.2 Member Typedef Documentation

**10.104.2.1 template<typename I, typename G, typename N> typedef N::complex_fwd_iter
 mln::complex_neighborhood_fwd_piter< I, G, N >::iter_type**

The type of the underlying complex iterator.

Definition at line 64 of file `complex_neighborhood_piter.hh`.

**10.104.2.2 template<typename I, typename G, typename N> typedef N ::psite
 mln::complex_neighborhood_fwd_piter< I, G, N >::psite**

The [Pseudo_Site](#) type.

Definition at line 62 of file `complex_neighborhood_piter.hh`.

10.104.3 Constructor & Destructor Documentation

**10.104.3.1 template<typename I , typename G , typename N > mln::complex_-
 neighborhood_fwd_piter< I, G, N >::complex_neighborhood_fwd_piter ()
 [inline]**

Construction.

Definition at line 198 of file complex_neighborhood_piter.hh.

10.104.4 Member Function Documentation

10.104.4.1 `template<typename I , typename G , typename N > const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter () const [inline]`

Accessors.

Definition at line 275 of file complex_neighborhood_piter.hh.

10.104.4.2 `void mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.105 mln::complex_psite< D, G > Class Template Reference

[Point](#) site associated to a [mln::p_complex](#).

```
#include <complex_psite.hh>
```

Inherits `pseudo_site_base_< const G::site &, complex_psite< D, G > >`.

Public Member Functions

- [complex_psite](#) ()
Construction and assignment.
- [complex_psite](#) (const [p_complex](#)< D, G > &pc, const [topo::face](#)< D > &face)
- bool [is_valid](#) () const
Psite manipulators.
- void [invalidate](#) ()
Invalidate this psite.
- const [target](#) & [site_set](#) () const
Site set manipulators.
- void [change_target](#) (const [target](#) &new_target)
Set the target site_set.

- const [topo::face](#)< D > & [face](#) () const
Face handle manipulators.
- unsigned [n](#) () const
Return the dimension of the face of this psite.
- unsigned [face_id](#) () const
Return the id of the face of this psite.

10.105.1 Detailed Description

template<unsigned D, typename G> class mln::complex_psite< D, G >

[Point](#) site associated to a [mln::p_complex](#).

Template Parameters

- D* The dimension of the complex this psite belongs to.
G The geometry of the complex.

Definition at line 60 of file `complex_psite.hh`.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite () [inline]

Construction and assignment.

Definition at line 203 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::invalidate()`.

10.105.2.2 template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite (const [p_complex](#)< D, G > & *pc*, const [topo::face](#)< D > & *face*) [inline]

Precondition

`pc.cplx() == face.cplx()`.

Definition at line 211 of file `complex_psite.hh`.

References `mln::topo::face< D >::cplx()`, `mln::p_complex< D, G >::cplx()`, and `mln::complex_psite< D, G >::is_valid()`.

10.105.3 Member Function Documentation

10.105.3.1 template<unsigned D, typename G > void mln::complex_psite< D, G >::change_target (const *target* & *new_target*) [inline]

Set the target site_set.

Definition at line 280 of file complex_psite.hh.

References mln::p_complex< D, G >::cplx(), and mln::complex_psite< D, G >::invalidate().

10.105.3.2 `template<unsigned D, typename G > const topo::face< D > & mln::complex_psite< D, G >::face () const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 301 of file complex_psite.hh.

Referenced by mln::operator!=(), and mln::operator==().

10.105.3.3 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::face_id () const [inline]`

Return the id of the face of this psite.

Definition at line 317 of file complex_psite.hh.

Referenced by mln::complex_image< D, G, V >::operator()().

10.105.3.4 `template<unsigned D, typename G > void mln::complex_psite< D, G >::invalidate () [inline]`

Invalidate this psite.

Definition at line 251 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::change_target(), and mln::complex_psite< D, G >::complex_psite().

10.105.3.5 `template<unsigned D, typename G > bool mln::complex_psite< D, G >::is_valid () const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 239 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::complex_psite(), and mln::p_complex< D, G >::has().

10.105.3.6 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::n () const [inline]`

Return the dimension of the face of this psite.

Definition at line 309 of file complex_psite.hh.

Referenced by mln::make::cell(), and mln::complex_image< D, G, V >::operator()().

10.105.3.7 `template<unsigned D, typename G > const p_complex< D, G > & mln::complex_psite< D, G >::site_set () const [inline]`

[Site](#) set manipulators.

Return the [mln::p_complex](#) this site is built on. (shortcut for *target()).

Precondition

Member `face_` is valid.

Definition at line 259 of file `complex_psite.hh`.

Referenced by `mln::p_complex< D, G >::has()`, `mln::operator!=()`, and `mln::operator==()`.

10.106 mln::complex_window_bkd_piter< I, G, W > Class Template Reference

Backward iterator on complex window.

`#include <complex_window_piter.hh>`

Inherits `site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >`.

Public Types

- `typedef W::complex_bkd_iter iter_type`
The type of the underlying complex iterator.
- `typedef W::psite psite`
The [Pseudo_Site](#) type.

Public Member Functions

- `void next ()`
Go to the next element.
- `complex_window_bkd_piter ()`
Construction.
- `const iter_type & iter () const`
Accessors.

10.106.1 Detailed Description

`template<typename I, typename G, typename W> class mln::complex_window_bkd_piter< I, G, W >`

Backward iterator on complex window.

Definition at line 124 of file complex_window_piter.hh.

10.106.2 Member Typedef Documentation

10.106.2.1 `template<typename I, typename G, typename W> typedef W::complex_bkd_iter
mln::complex_window_bkd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file complex_window_piter.hh.

10.106.2.2 `template<typename I, typename G, typename W> typedef W ::psite
mln::complex_window_bkd_piter< I, G, W >::psite`

The [Pseudo_Site](#) type.

Definition at line 133 of file complex_window_piter.hh.

10.106.3 Constructor & Destructor Documentation

10.106.3.1 `template<typename I , typename G , typename W > mln::complex_
window_bkd_piter< I, G, W >::complex_window_bkd_piter ()
[inline]`

Construction.

Definition at line 305 of file complex_window_piter.hh.

10.106.4 Member Function Documentation

10.106.4.1 `template<typename I , typename G , typename W > const W::complex_bkd_iter &
mln::complex_window_bkd_piter< I, G, W >::iter () const [inline]`

Accessors.

Definition at line 384 of file complex_window_piter.hh.

10.106.4.2 `void mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.107 mln::complex_window_fwd_piter< I, G, W > Class Template Reference

Forward iterator on complex window.

```
#include <complex_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >`.

Public Types

- typedef `W::complex_fwd_iter` [iter_type](#)
The type of the underlying complex iterator.
- typedef `W::psite` [psite](#)
The [Pseudo_Site](#) type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_window_fwd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.107.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_fwd_piter< I, G, W >
```

Forward iterator on complex window.

Definition at line 54 of file `complex_window_piter.hh`.

10.107.2 Member Typedef Documentation

10.107.2.1 `template<typename I, typename G, typename W> typedef W::complex_fwd_iter mln::complex_window_fwd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 65 of file `complex_window_piter.hh`.

10.107.2.2 `template<typename I, typename G, typename W> typedef W ::psite
mln::complex_window_fwd_piter< I, G, W >::psite`

The [Pseudo_Site](#) type.

Definition at line 63 of file complex_window_piter.hh.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 `template<typename I , typename G , typename W > mln::complex_
window_fwd_piter< I, G, W >::complex_window_fwd_piter ()
[inline]`

Construction.

Definition at line 197 of file complex_window_piter.hh.

10.107.4 Member Function Documentation

10.107.4.1 `template<typename I , typename G , typename W > const W::complex_fwd_iter &
mln::complex_window_fwd_piter< I, G, W >::iter () const [inline]`

Accessors.

Definition at line 275 of file complex_window_piter.hh.

10.107.4.2 `void mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.108 mln::decorated_image< I, D > Struct Template Reference

[Image](#) that can have additional features.

```
#include <decorated_image.hh>
```

Inherits decorated_image_impl< I, decorated_image< I, D > >, and image_identity< I, I::domain_t, decorated_image< I, D > >.

Public Types

- typedef [impl_::lvalue](#) lvalue

Return type of read-write access.

- typedef I::psite [psite](#)

Type of the psite.

- typedef I::rvalue [rvalue](#)

Return type of read-only access.

- typedef [decorated_image](#)< tag::image_< I >, tag::data_< D > > [skeleton](#)

Skeleton.

Public Member Functions

- [decorated_image](#) ()

Ctors.

- D & [decoration](#) ()

Give the decoration.

- const D & [decoration](#) () const

Give the decoration.

- [operator decorated_image](#)< const I, D > () const

Const promotion via conversion.

- [rvalue operator](#)() (const [psite](#) &p) const

Read-only access of pixel value at point site p.

- [lvalue operator](#)() (const [psite](#) &p)

Read-write access of pixel value at point site p.

10.108.1 Detailed Description

`template<typename I, typename D> struct mln::decorated_image< I, D >`

[Image](#) that can have additional features.

Definition at line 81 of file `decorated_image.hh`.

10.108.2 Member Typedef Documentation

10.108.2.1 `template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue`

Return type of read-write access.

Definition at line 95 of file `decorated_image.hh`.

10.108.2.2 template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite

Type of the psite.

Definition at line 90 of file decorated_image.hh.

10.108.2.3 template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue

Return type of read-only access.

Definition at line 93 of file decorated_image.hh.

10.108.2.4 template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton

Skeleton.

Definition at line 108 of file decorated_image.hh.

10.108.3 Constructor & Destructor Documentation**10.108.3.1 template<typename I , typename D > mln::decorated_image< I, D >::decorated_image () [inline]**

Ctors.

Definition at line 161 of file decorated_image.hh.

10.108.4 Member Function Documentation**10.108.4.1 template<typename I , typename D > const D & mln::decorated_image< I, D >::decoration () const [inline]**

Give the decoration.

Definition at line 249 of file decorated_image.hh.

10.108.4.2 template<typename I , typename D > D & mln::decorated_image< I, D >::decoration () [inline]

Give the decoration.

Definition at line 257 of file decorated_image.hh.

10.108.4.3 template<typename I , typename D > mln::decorated_image< I, D >::operator decorated_image< const I, D > () const [inline]

Const promotion via conversion.

Definition at line 239 of file decorated_image.hh.

10.108.4.4 `template<typename I , typename D > decorated_image< I, D >::rvalue
mln::decorated_image< I, D >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 197 of file decorated_image.hh.

10.108.4.5 `template<typename I , typename D > decorated_image< I, D >::lvalue
mln::decorated_image< I, D >::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site *p*.

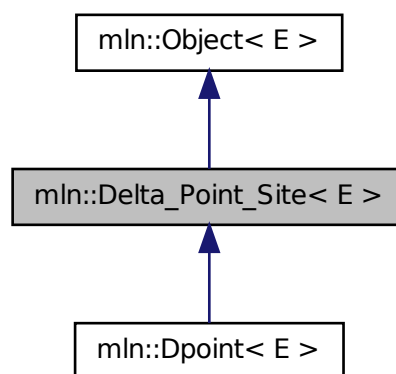
Definition at line 206 of file decorated_image.hh.

10.109 mln::Delta_Point_Site< E > Struct Template Reference

FIXME: Doc!

```
#include <delta_point_site.hh>
```

Inheritance diagram for mln::Delta_Point_Site< E >:



10.109.1 Detailed Description

`template<typename E> struct mln::Delta_Point_Site< E >`

FIXME: Doc!

Definition at line 79 of file delta_point_site.hh.

10.110 mln::Delta_Point_Site< void > Struct Template Reference

Delta point site category flag type.

```
#include <delta_point_site.hh>
```

10.110.1 Detailed Description

template<> struct mln::Delta_Point_Site< void >

Delta point site category flag type.

Definition at line 70 of file delta_point_site.hh.

10.111 mln::doc::Accumulator< E > Struct Template Reference

Documentation class for [mln::Accumulator](#).

```
#include <accumulator.hh>
```

Public Types

- typedef void [argument](#)
The argument type of elements to accumulate.

Public Member Functions

- void [init](#) ()
Initialize the accumulator.
- void [take](#) (const E &other)
Take into account another accumulator `other`.
- void [take](#) (const [argument](#) &t)
Take into account a argument `t` (an element).

10.111.1 Detailed Description

template<typename E> struct mln::doc::Accumulator< E >

Documentation class for [mln::Accumulator](#).

See also

[mln::Accumulator](#)

Definition at line 36 of file doc/accumulator.hh.

10.111.2 Member Typedef Documentation

10.111.2.1 `template<typename E> typedef void mln::doc::Accumulator< E >::argument`

The argument type of elements to accumulate.

Definition at line 39 of file doc/accumulator.hh.

10.111.3 Member Function Documentation

10.111.3.1 `template<typename E> void mln::doc::Accumulator< E >::init ()`

Initialize the accumulator.

10.111.3.2 `template<typename E> void mln::doc::Accumulator< E >::take (const E & other)`

Take into account another accumulator *other*.

10.111.3.3 `template<typename E> void mln::doc::Accumulator< E >::take (const argument & t)`

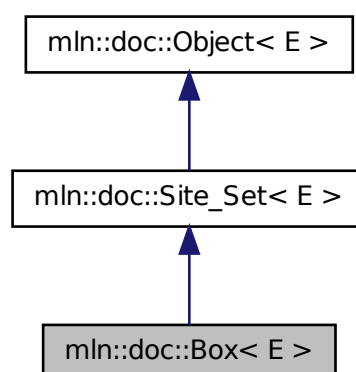
Take into account a argument *t* (an element).

10.112 `mln::doc::Box< E >` Struct Template Reference

Documentation class for `mln::Box`.

```
#include <box.hh>
```

Inheritance diagram for `mln::doc::Box< E >`:



Public Types

- typedef void [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef void [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef void [psite](#)
PSite associated type.
- typedef void [site](#)
[Site](#) associated type.

Public Member Functions

- const E & [bbox](#) () const
Return the bounding box of this point set.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this site set.
- unsigned [nsites](#) () const
Return the number of points of this box.
- const [site](#) & [pmax](#) () const
Give the box "maximum" point.
- const [site](#) & [pmin](#) () const
Give the box "minimum" point.

10.112.1 Detailed Description

template<typename E> struct mln::doc::Box< E >

Documentation class for [mln::Box](#).

See also

[mln::Box](#)

Definition at line 36 of file core/concept/doc/box.hh.

10.112.2 Member Typedef Documentation

10.112.2.1 template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter [inherited]

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file mln/core/concept/doc/site_set.hh.

10.112.2.2 `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`
`[inherited]`

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file mln/core/concept/doc/site_set.hh.

10.112.2.3 `template<typename E > typedef void mln::doc::Site_Set< E >::psite` `[inherited]`

PSite associated type.

Definition at line 45 of file mln/core/concept/doc/site_set.hh.

10.112.2.4 `template<typename E > typedef void mln::doc::Site_Set< E >::site` `[inherited]`

[Site](#) associated type.

Definition at line 41 of file mln/core/concept/doc/site_set.hh.

10.112.3 Member Function Documentation

10.112.3.1 `template<typename E > const E& mln::doc::Box< E >::bbox () const`

Return the bounding box of this point set.

Return the bounding box of this point set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.112.3.2 `template<typename E > bool mln::doc::Site_Set< E >::has (const psite & p) const`
`[inherited]`

Test if *p* belongs to this site set.

Parameters

`[in]` *p* A psite.

Returns

True if *p* is an element of the site set.

10.112.3.3 `template<typename E > unsigned mln::doc::Box< E >::nsites () const`

Return the number of points of this box.

Return the number of points of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.112.3.4 template<typename E > const site& mln::doc::Box< E >::pmax () const

Give the box "maximum" point.

Return the "maximum" point w.r.t. the ordering between points. For instance, with [mln::box2d](#), this maximum is the bottom right point of the box.

10.112.3.5 template<typename E > const site& mln::doc::Box< E >::pmin () const

Give the box "minimum" point.

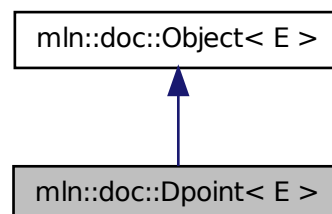
Return the "minimum" point w.r.t. the ordering between points. For instance, with [mln::box2d](#), this minimum is the top left point of the box.

10.113 mln::doc::Dpoint< E > Struct Template Reference

Documentation class for [mln::Dpoint](#).

```
#include <dpoint.hh>
```

Inheritance diagram for mln::doc::Dpoint< E >:

**Public Types**

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [point](#)
Site associated type.

Public Member Functions

- [coord operator\[\]](#) (unsigned i) const
Read-only access to the i -th coordinate value.

10.113.1 Detailed Description

template<typename E> struct mln::doc::Dpoint< E >

Documentation class for [mln::Dpoint](#).

See also

[mln::Dpoint](#)

Definition at line 36 of file concept/doc/dpoint.hh.

10.113.2 Member Typedef Documentation

10.113.2.1 template<typename E > typedef void mln::doc::Dpoint< E >::coord

Coordinate associated type.

Definition at line 56 of file concept/doc/dpoint.hh.

10.113.2.2 template<typename E > typedef void mln::doc::Dpoint< E >::dpoint

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 52 of file concept/doc/dpoint.hh.

10.113.2.3 template<typename E > typedef void mln::doc::Dpoint< E >::point

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 47 of file concept/doc/dpoint.hh.

10.113.3 Member Enumeration Documentation

10.113.3.1 template<typename E > anonymous enum

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 42 of file concept/doc/dpoint.hh.

10.113.4 Member Function Documentation**10.113.4.1** `template<typename E> coord mln::doc::Dpoint< E >::operator[] (unsigned i) const`

Read-only access to the `i`-th coordinate value.

Parameters

[in] `i` The coordinate index.

Precondition

`i < dim`

Returns

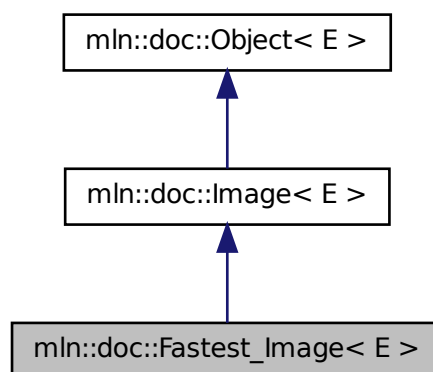
The value of the `i`-th coordinate.

10.114 mln::doc::Fastest_Image< E > Struct Template Reference

Documentation class for the concept of images that have the speed property set to "fastest".

```
#include <image_fastest.hh>
```

Inheritance diagram for `mln::doc::Fastest_Image< E >`:



Public Types

- typedef void [bkd_piter](#)
Backward point iterator associated type.
- typedef void [coord](#)
Coordinate associated type.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_piter](#)
Forward point iterator associated type.
- typedef void [lvalue](#)
Type returned by the read-write pixel value operator.
- typedef void [point](#)
Site associated type.
- typedef void [pset](#)
Point set associated type.
- typedef void [psite](#)
Point_Site associated type.
- typedef void [rvalue](#)
Type returned by the read pixel value operator.
- typedef void [skeleton](#)
Associate type that describes how this type of image is constructed.
- typedef void [value](#)
Value associated type.
- typedef void [vset](#)
Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- unsigned [border](#) ()
Give the border thickness.
- const [value](#) * [buffer](#) () const
Give a hook to the value buffer.

- `int delta_index (const dpoint &dp)`
Give the offset corresponding to the delta-point `dp`.
- `const pset & domain () const`
Give the definition domain of the image.
- `bool has (const psite &p) const`
Test if the image owns the point site `p`.
- `bool has (const psite &p) const`
Test if `p` belongs to the image domain.
- `bool is_valid () const`
Test if the image have been initialized.
- `unsigned nelements () const`
Give the number of pixels of the image including those of the virtual border.
- `unsigned nsites () const`
Give the number of points of the image domain.
- `lvalue operator() (const psite &p)`
Read-write access to the image value located at `p`.
- `rvalue operator() (const psite &p) const`
Read-only access to the image value located at `p`.
- `rvalue operator[] (unsigned o) const`
Read-only access to the image value at offset `o`.
- `lvalue operator[] (unsigned o)`
Read-write access to the image value at offset `o`.
- `point point_at_index (unsigned o) const`
Give the point at offset `o`.
- `const vset & values () const`
Give the set of values of the image.

10.114.1 Detailed Description

`template<typename E> struct mln::doc::Fastest_Image< E >`

Documentation class for the concept of images that have the speed property set to "fastest".

Definition at line 36 of file `concept/doc/image_fastest.hh`.

10.114.2 Member Typedef Documentation

10.114.2.1 `template<typename E > typedef void mln::doc::Image< E >::bkd_piter` `[inherited]`

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 147 of file core/concept/doc/image.hh.

10.114.2.2 `template<typename E > typedef void mln::doc::Image< E >::coord` `[inherited]`

Coordinate associated type.

Definition at line 131 of file core/concept/doc/image.hh.

10.114.2.3 `template<typename E > typedef void mln::doc::Image< E >::dpoint` `[inherited]`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

10.114.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter` `[inherited]`

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

10.114.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue` `[inherited]`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.114.2.6 `template<typename E > typedef void mln::doc::Image< E >::point` `[inherited]`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.114.2.7 template<typename E > typedef void mln::doc::Image< E >::pset [inherited]

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.114.2.8 template<typename E > typedef void mln::doc::Image< E >::psite [inherited]

[Point_Site](#) associated type.

Invariant

This type has to derive from mln::Point_Site.

Definition at line 126 of file core/concept/doc/image.hh.

10.114.2.9 template<typename E > typedef void mln::doc::Image< E >::rvalue [inherited]

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

**10.114.2.10 template<typename E > typedef void mln::doc::Image< E >::skeleton
 [inherited]**

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.114.2.11 template<typename E > typedef void mln::doc::Image< E >::value [inherited]

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.114.2.12 template<typename E > typedef void mln::doc::Image< E >::vset [inherited]

[Value](#) set associated type.

Invariant

This type has to derive from mln::Value_Set.

Definition at line 57 of file core/concept/doc/image.hh.

10.114.3 Member Function Documentation

10.114.3.1 `template<typename E> const box<point>& mln::doc::Image< E>::bbox () const`
[inherited]

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

Returns

A bounding box of the image domain.

10.114.3.2 `template<typename E> unsigned mln::doc::Fastest_Image< E>::border ()`

Give the border thickness.

Precondition

The image has to be initialized.

10.114.3.3 `template<typename E> const value* mln::doc::Fastest_Image< E>::buffer () const`

Give a hook to the value buffer.

Precondition

The image has to be initialized.

10.114.3.4 `template<typename E> int mln::doc::Fastest_Image< E>::delta_index (const
dpoint & dp)`

Give the offset corresponding to the delta-point *dp*.

Parameters

[in] *dp* A delta-point.

Precondition

The image has to be initialized.

10.114.3.5 `template<typename E> const pset& mln::doc::Image< E>::domain () const`
[inherited]

Give the definition domain of the image.

Returns

A reference to the domain point set.

10.114.3.6 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const`
[*inherited*]

Test if the image owns the point site *p*.

Returns

True if accessing the image value at *p* is possible, that is, does not abort the execution.

10.114.3.7 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const`
[*inherited*]

Test if *p* belongs to the image domain.

Parameters

[*in*] *p* A point site.

Returns

True if *p* belongs to the image domain.

Invariant

has(*p*) is true => has(*p*) is also true.

10.114.3.8 `template<typename E> bool mln::doc::Image< E >::is_valid () const`
[*inherited*]

Test if the image have been initialized.

10.114.3.9 `template<typename E> unsigned mln::doc::Fastest_Image< E >::nelements () const`

Give the number of pixels of the image including those of the virtual border.

Precondition

The image has to be initialized.

10.114.3.10 `template<typename E> unsigned mln::doc::Image< E >::nsites () const`
[*inherited*]

Give the number of points of the image domain.

10.114.3.11 `template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & p)`
[*inherited*]

Read-write access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (assignable).

10.114.3.12 `template<typename E> rvalue mln::doc::Image< E >::operator() (const psite & p) const` **[inherited]**

Read-only access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (not assignable).

10.114.3.13 `template<typename E> rvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o) const`

Read-only access to the image value at offset *o*.

Parameters

[in] *o* An offset.

Precondition

o < [nelements\(\)](#)

Returns

The value at *o* (not assignable).

10.114.3.14 `template<typename E> lvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o)`

Read-write access to the image value at offset *o*.

Parameters

[in] *o* An offset.

Precondition

$o < \text{nelements}()$

Returns

The value at o (assignable).

10.114.3.15 `template<typename E> point mln::doc::Fastest_Image< E >::point_at_index (unsigned o) const`

Give the point at offset o .

Parameters

[in] o An offset.

Precondition

The image has to be initialized.

$o < \text{nelements}()$

10.114.3.16 `template<typename E> const vset& mln::doc::Image< E >::values () const [inherited]`

Give the set of values of the image.

Returns

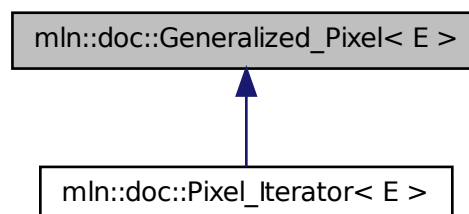
A reference to the value set.

10.115 mln::doc::Generalized_Pixel< E > Struct Template Reference

Documentation class for [mln::Generalized_Pixel](#).

```
#include <generalized_pixel.hh>
```

Inheritance diagram for mln::doc::Generalized_Pixel< E >:



Public Types

- typedef void [image](#)
Image associated type (with possible const qualification).
- typedef void [rvalue](#)
Read-only value associated type.
- typedef void [value](#)
Value associated type.

Public Member Functions

- [image](#) & [ima](#) () const
Give the image of this generalized pixel.
- [rvalue](#) [val](#) () const
Give the value of this generalized pixel.

10.115.1 Detailed Description

template<typename E> struct mln::doc::Generalized_Pixel< E >

Documentation class for [mln::Generalized_Pixel](#).

See also

[mln::Generalized_Pixel](#)

Definition at line 45 of file doc/generalized_pixel.hh.

10.115.2 Member Typedef Documentation

10.115.2.1 **template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image**

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized_pixel.hh.

10.115.2.2 **template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue**

Read-only value associated type.

Definition at line 55 of file doc/generalized_pixel.hh.

10.115.2.3 **template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value**

[Value](#) associated type.

Definition at line 52 of file doc/generalized_pixel.hh.

10.115.3 Member Function Documentation

10.115.3.1 `template<typename E> image& mln::doc::Generalized_Pixel< E >::ima () const`

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

10.115.3.2 `template<typename E> rvalue mln::doc::Generalized_Pixel< E >::val () const`

Give the value of this generalized pixel.

Returns

A read-only value.

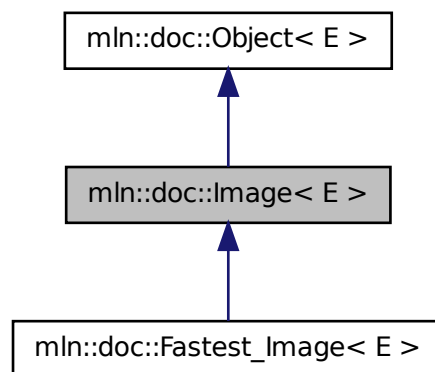
Reimplemented in [mln::doc::Pixel_Iterator< E >](#).

10.116 mln::doc::Image< E > Struct Template Reference

Documentation class for [mln::Image](#).

```
#include <image.hh>
```

Inheritance diagram for mln::doc::Image< E >:



Public Types

- typedef void [bkd_piter](#)
Backward point iterator associated type.
- typedef void [coord](#)

Coordinate associated type.

- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_piter](#)
Forward point iterator associated type.
- typedef void [lvalue](#)
Type returned by the read-write pixel value operator.
- typedef void [point](#)
Site associated type.
- typedef void [pset](#)
Point set associated type.
- typedef void [psite](#)
Point_Site associated type.
- typedef void [rvalue](#)
Type returned by the read pixel value operator.
- typedef void [skeleton](#)
Associate type that describes how this type of image is constructed.
- typedef void [value](#)
Value associated type.
- typedef void [vset](#)
Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- const [pset](#) & [domain](#) () const
Give the definition domain of the image.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the point site p.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the image domain.
- bool [is_valid](#) () const
Test if the image have been initialized.

- unsigned [nsites](#) () const
Give the number of points of the image domain.
- [rvalue operator\(\)](#) (const [psite](#) &p) const
Read-only access to the image value located at p.
- [lvalue operator\(\)](#) (const [psite](#) &p)
Read-write access to the image value located at p.
- const [vset](#) & [values](#) () const
Give the set of values of the image.

10.116.1 Detailed Description

template<typename E> struct mln::doc::Image< E >

Documentation class for [mln::Image](#).

See also

[mln::Image](#)

Definition at line 36 of file core/concept/doc/image.hh.

10.116.2 Member Typedef Documentation

10.116.2.1 template<typename E > typedef void mln::doc::Image< E >::bkd_piter

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 147 of file core/concept/doc/image.hh.

10.116.2.2 template<typename E > typedef void mln::doc::Image< E >::coord

Coordinate associated type.

Definition at line 131 of file core/concept/doc/image.hh.

10.116.2.3 template<typename E > typedef void mln::doc::Image< E >::dpoint

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

10.116.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

10.116.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.116.2.6 `template<typename E > typedef void mln::doc::Image< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.116.2.7 `template<typename E > typedef void mln::doc::Image< E >::pset`

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.116.2.8 `template<typename E > typedef void mln::doc::Image< E >::psite`

[Point_Site](#) associated type.

Invariant

This type has to derive from [mln::Point_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

10.116.2.9 `template<typename E > typedef void mln::doc::Image< E >::rvalue`

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

10.116.2.10 template<typename E > typedef void mln::doc::Image< E >::skeleton

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.116.2.11 template<typename E > typedef void mln::doc::Image< E >::value

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.116.2.12 template<typename E > typedef void mln::doc::Image< E >::vset

[Value](#) set associated type.

Invariant

This type has to derive from mln::Value_Set.

Definition at line 57 of file core/concept/doc/image.hh.

10.116.3 Member Function Documentation**10.116.3.1 template<typename E > const box<point>& mln::doc::Image< E >::bbox () const**

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

Returns

A bounding box of the image domain.

10.116.3.2 template<typename E > const pset& mln::doc::Image< E >::domain () const

Give the definition domain of the image.

Returns

A reference to the domain point set.

10.116.3.3 template<typename E > bool mln::doc::Image< E >::has (const psite & p) const

Test if the image owns the point site p.

Returns

True if accessing the image value at p is possible, that is, does not abort the execution.

10.116.3.4 template<typename E> bool mln::doc::Image< E>::has (const psite & p) const

Test if *p* belongs to the image domain.

Parameters

[in] *p* A point site.

Returns

True if *p* belongs to the image domain.

Invariant

has(*p*) is true => has(*p*) is also true.

10.116.3.5 template<typename E> bool mln::doc::Image< E>::is_valid () const

Test if the image have been initialized.

10.116.3.6 template<typename E> unsigned mln::doc::Image< E>::nsites () const

Give the number of points of the image domain.

10.116.3.7 template<typename E> rvalue mln::doc::Image< E>::operator() (const psite & p) const

Read-only access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (not assignable).

10.116.3.8 template<typename E> lvalue mln::doc::Image< E>::operator() (const psite & p)

Read-write access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (assignable).

10.116.3.9 `template<typename E> const vset& mln::doc::Image< E >::values () const`

Give the set of values of the image.

Returns

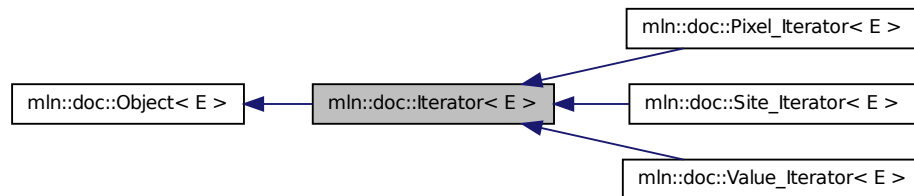
A reference to the value set.

10.117 `mln::doc::Iterator< E >` Struct Template Reference

Documentation class for [mln::Iterator](#).

```
#include <iterator.hh>
```

Inheritance diagram for `mln::doc::Iterator< E >`:

**Public Member Functions**

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- void [start](#) ()
Start an iteration.

10.117.1 Detailed Description

```
template<typename E> struct mln::doc::Iterator< E >
```

Documentation class for [mln::Iterator](#).

See also

[mln::Iterator](#)

Definition at line 36 of file `doc/iterator.hh`.

10.117.2 Member Function Documentation

10.117.2.1 `template<typename E> void mln::doc::Iterator< E >::invalidate ()`

Invalidate the iterator.

10.117.2.2 `template<typename E> bool mln::doc::Iterator< E >::is_valid () const`

Returns true if the iterator is valid, that is, designates an element.

10.117.2.3 `template<typename E> void mln::doc::Iterator< E >::start ()`

Start an iteration.

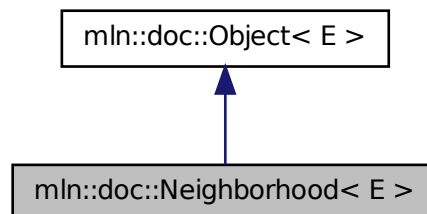
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.118 `mln::doc::Neighborhood< E >` Struct Template Reference

Documentation class for [mln::Neighborhood](#).

```
#include <neighborhood.hh>
```

Inheritance diagram for `mln::doc::Neighborhood< E >`:



Public Types

- typedef void [bkd_niter](#)
Site_iterator type associated to this neighborhood to browse neighbors in a backward way.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_niter](#)
Site_iterator type associated to this neighborhood to browse neighbors in a forward way.

- typedef void [niter](#)
[Site_Iterator](#) type associated to this neighborhood to browse neighbors.
- typedef void [point](#)
[Site](#) associated type.

10.118.1 Detailed Description

template<typename E> struct mln::doc::Neighborhood< E >

Documentation class for [mln::Neighborhood](#).

See also

[mln::Neighborhood](#)

Definition at line 37 of file core/concept/doc/neighborhood.hh.

10.118.2 Member Typedef Documentation

10.118.2.1 template<typename E > typedef void mln::doc::Neighborhood< E >::bkd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a backward way.

Definition at line 52 of file core/concept/doc/neighborhood.hh.

10.118.2.2 template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint

Dpsite associated type.

Definition at line 55 of file core/concept/doc/neighborhood.hh.

10.118.2.3 template<typename E > typedef void mln::doc::Neighborhood< E >::fwd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a forward way.

Definition at line 47 of file core/concept/doc/neighborhood.hh.

10.118.2.4 template<typename E > typedef void mln::doc::Neighborhood< E >::niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors.

Definition at line 42 of file core/concept/doc/neighborhood.hh.

10.118.2.5 template<typename E > typedef void mln::doc::Neighborhood< E >::point

[Site](#) associated type.

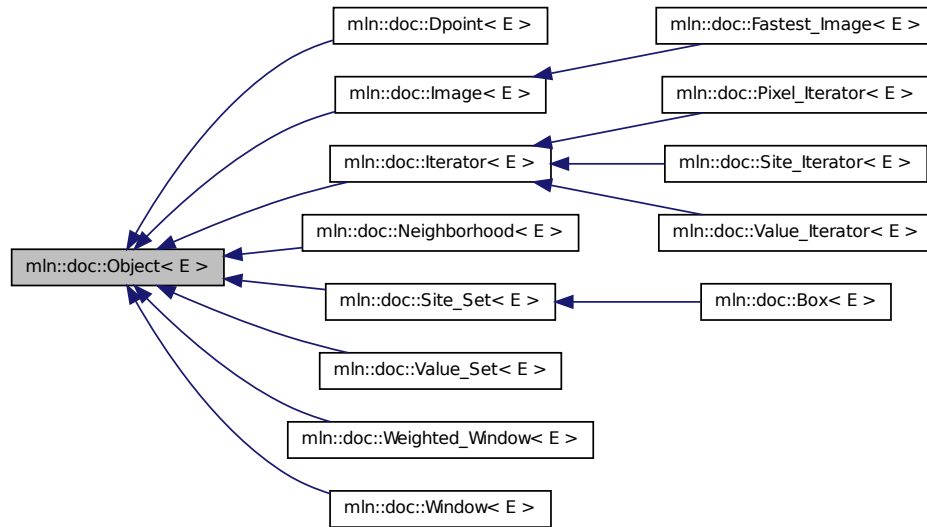
Definition at line 58 of file core/concept/doc/neighborhood.hh.

10.119 mln::doc::Object< E > Struct Template Reference

Documentation class for [mln::Object](#).

```
#include <object.hh>
```

Inheritance diagram for mln::doc::Object< E >:



10.119.1 Detailed Description

```
template<typename E> struct mln::doc::Object< E >
```

Documentation class for [mln::Object](#).

See also

[mln::Object](#)

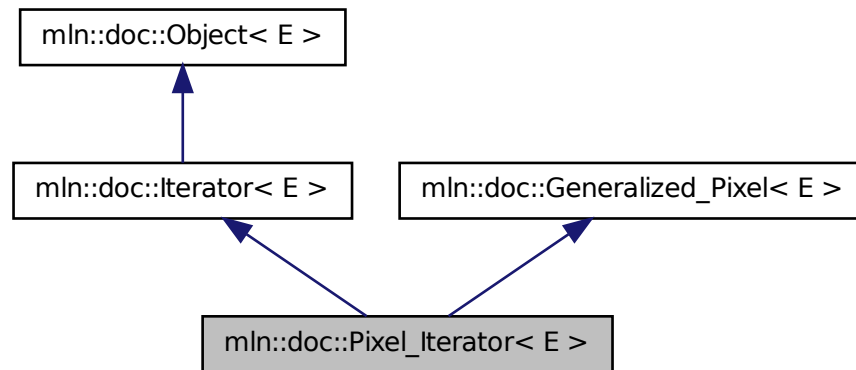
Definition at line 46 of file doc/object.hh.

10.120 mln::doc::Pixel_Iterator< E > Struct Template Reference

Documentation class for [mln::Iterator](#).

```
#include <pixel_iterator.hh>
```

Inheritance diagram for mln::doc::Pixel_Iterator< E >:



Public Types

- typedef void [image](#)
Image associated type (with possible const qualification).
- typedef void [lvalue](#)
Type returned by the read-write dereference operator.
- typedef void [rvalue](#)
Read-only value associated type.
- typedef void [value](#)
Value associated type.

Public Member Functions

- [image](#) & [ima](#) () const
Give the image of this generalized pixel.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- void [start](#) ()
Start an iteration.

- [lvalue val](#) () const

Give the pixel value.

10.120.1 Detailed Description

template<typename E> struct mln::doc::Pixel_Iterator< E >

Documentation class for [mln::Iterator](#).

See also

mln::Pixel_Iterator

Definition at line 36 of file doc/pixel_iterator.hh.

10.120.2 Member Typedef Documentation

**10.120.2.1 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image
[inherited]**

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized_pixel.hh.

10.120.2.2 template<typename E > typedef void mln::doc::Pixel_Iterator< E >::lvalue

Type returned by the read-write dereference operator.

Definition at line 41 of file doc/pixel_iterator.hh.

**10.120.2.3 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue
[inherited]**

Read-only value associated type.

Definition at line 55 of file doc/generalized_pixel.hh.

**10.120.2.4 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value
[inherited]**

[Value](#) associated type.

Definition at line 52 of file doc/generalized_pixel.hh.

10.120.3 Member Function Documentation

**10.120.3.1 template<typename E > image& mln::doc::Generalized_Pixel< E >::ima () const
[inherited]**

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

10.120.3.2 `template<typename E > void mln::doc::Iterator< E >::invalidate ()`
[*inherited*]

Invalidate the iterator.

10.120.3.3 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const`
[*inherited*]

Returns true if the iterator is valid, that is, designates an element.

10.120.3.4 `template<typename E > void mln::doc::Iterator< E >::start ()` [*inherited*]

Start an iteration.

Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.120.3.5 `template<typename E > lvalue mln::doc::Pixel_Iterator< E >::val () const`

Give the pixel value.

Returns

The current pixel value; this value cannot be modified.

Reimplemented from [mln::doc::Generalized_Pixel< E >](#).

10.121 mln::doc::Point_Site< E > Struct Template Reference

Documentation class for mln::Point_Site.

```
#include <point_site.hh>
```

Public Types

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [mesh](#)
Mesh associated type.
- typedef void [point](#)
Site associated type.

Public Member Functions

- [coord operator\[\]](#) (unsigned i) const
Read-only access to the i -th coordinate value.
- const [point](#) & [to_point](#)() const
Give a reference to the corresponding point.

10.121.1 Detailed Description

template<typename E> struct mln::doc::Point_Site< E >

Documentation class for mln::Point_Site.

See also

[mln::Point_Site](#)

Definition at line 37 of file doc/point_site.hh.

10.121.2 Member Typedef Documentation

10.121.2.1 template<typename E > typedef void mln::doc::Point_Site< E >::coord

Coordinate associated type.

Definition at line 62 of file doc/point_site.hh.

10.121.2.2 template<typename E > typedef void mln::doc::Point_Site< E >::dpoint

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 58 of file doc/point_site.hh.

10.121.2.3 template<typename E > typedef void mln::doc::Point_Site< E >::mesh

[Mesh](#) associated type.

Invariant

This type has to derive from [mln::Mesh](#).

Definition at line 48 of file doc/point_site.hh.

10.121.2.4 `template<typename E > typedef void mln::doc::Point_Site< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 53 of file doc/point_site.hh.

10.121.3 Member Enumeration Documentation

10.121.3.1 `template<typename E > anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 43 of file doc/point_site.hh.

10.121.4 Member Function Documentation

10.121.4.1 `template<typename E > coord mln::doc::Point_Site< E >::operator[] (unsigned i) const`

Read-only access to the `i-th` coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

`i < dim`

Returns

The value of the `i-th` coordinate.

10.121.4.2 `template<typename E > const point& mln::doc::Point_Site< E >::to_point () const`

Give a reference to the corresponding point.

This method allows for iterators to refer to a point.

Returns

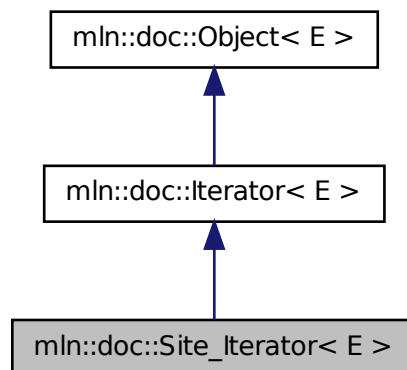
A point constant reference.

10.122 mln::doc::Site_Iterator< E > Struct Template Reference

Documentation class for [mln::Site_Iterator](#).

```
#include <point_iterator.hh>
```

Inheritance diagram for mln::doc::Site_Iterator< E >:



Public Types

- typedef void [psite](#)
[Point_Site](#) associated type.

Public Member Functions

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- [operator psite](#) () const
Conversion into a point-site.
- void [start](#) ()
Start an iteration.

10.122.1 Detailed Description

template<typename E> struct mln::doc::Site_Iterator< E >

Documentation class for [mln::Site_Iterator](#).

See also

[mln::Site_Iterator](#)

Definition at line 37 of file point_iterator.hh.

10.122.2 Member Typedef Documentation

10.122.2.1 template<typename E > typedef void mln::doc::Site_Iterator< E >::psite

[Point_Site](#) associated type.

Invariant

This type has to derive from mln::Point_Site.

Definition at line 43 of file point_iterator.hh.

10.122.3 Member Function Documentation

10.122.3.1 template<typename E > void mln::doc::Iterator< E >::invalidate ()
[inherited]

Invalidate the iterator.

10.122.3.2 template<typename E > bool mln::doc::Iterator< E >::is_valid () const
[inherited]

Returns true if the iterator is valid, that is, designates an element.

10.122.3.3 template<typename E > mln::doc::Site_Iterator< E >::operator psite () const

Conversion into a point-site.

Returns

A point site.

10.122.3.4 template<typename E > void mln::doc::Iterator< E >::start () [inherited]

Start an iteration.

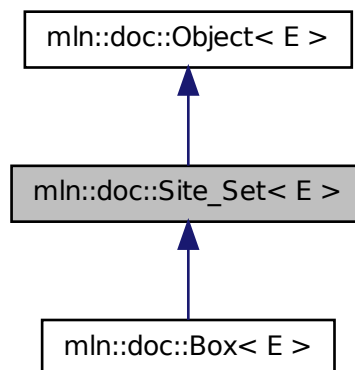
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.123 mln::doc::Site_Set< E > Struct Template Reference

Documentation class for [mln::Site_Set](#).

```
#include <site_set.hh>
```

Inheritance diagram for mln::doc::Site_Set< E >:



Public Types

- typedef void [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef void [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef void [psite](#)
PSite associated type.
- typedef void [site](#)
[Site](#) associated type.

Public Member Functions

- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this site set.*

10.123.1 Detailed Description

`template<typename E> struct mln::doc::Site_Set< E >`

Documentation class for [mln::Site_Set](#).

See also

[mln::Site_Set](#)

Definition at line 37 of file mln/core/concept/doc/site_set.hh.

10.123.2 Member Typedef Documentation

10.123.2.1 `template<typename E> typedef void mln::doc::Site_Set< E >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file mln/core/concept/doc/site_set.hh.

10.123.2.2 `template<typename E> typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file mln/core/concept/doc/site_set.hh.

10.123.2.3 `template<typename E> typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

Definition at line 45 of file mln/core/concept/doc/site_set.hh.

10.123.2.4 `template<typename E> typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

Definition at line 41 of file mln/core/concept/doc/site_set.hh.

10.123.3 Member Function Documentation

10.123.3.1 `template<typename E> bool mln::doc::Site_Set< E >::has (const psite & p) const`

Test if *p* belongs to this site set.

Parameters

[in] *p* A psite.

Returns

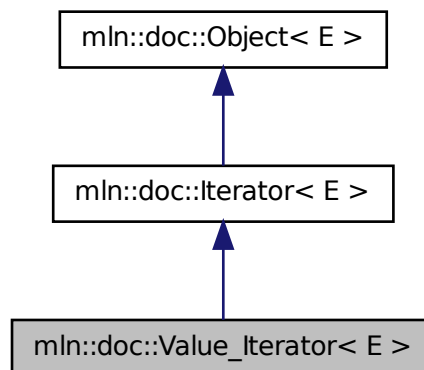
True if *p* is an element of the site set.

10.124 mln::doc::Value_Iterator< E > Struct Template Reference

Documentation class for mln::Value_Iterator.

```
#include <value_iterator.hh>
```

Inheritance diagram for mln::doc::Value_Iterator< E >:



Public Types

- typedef void [value](#)
Value associated type.

Public Member Functions

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- [operator value](#) () const
Conversion into a value.
- void [start](#) ()
Start an iteration.

10.124.1 Detailed Description

template<typename E> struct mln::doc::Value_Iterator< E >

Documentation class for mln::Value_Iterator.

See also

mln::Value_Iterator

Definition at line 37 of file doc/value_iterator.hh.

10.124.2 Member Typedef Documentation

10.124.2.1 template<typename E > typedef void mln::doc::Value_Iterator< E >::value

[Value](#) associated type.

Definition at line 41 of file doc/value_iterator.hh.

10.124.3 Member Function Documentation

**10.124.3.1 template<typename E > void mln::doc::Iterator< E >::invalidate ()
[inherited]**

Invalidate the iterator.

**10.124.3.2 template<typename E > bool mln::doc::Iterator< E >::is_valid () const
[inherited]**

Returns true if the iterator is valid, that is, designates an element.

10.124.3.3 template<typename E > mln::doc::Value_Iterator< E >::operator value () const

Conversion into a value.

Returns

A value.

10.124.3.4 template<typename E > void mln::doc::Iterator< E >::start () [inherited]

Start an iteration.

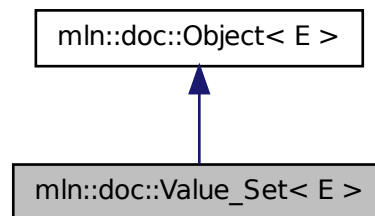
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.125 mln::doc::Value_Set< E > Struct Template Reference

Documentation class for mln::Value_Set.

```
#include <value_set.hh>
```

Inheritance diagram for `mln::doc::Value_Set< E >`:



Public Types

- typedef void `bkd_viter`
Backward [Value_Iterator](#) associated type.
- typedef void `fwd_viter`
Forward [Value_Iterator](#) associated type.
- typedef void `value`
[Value](#) associated type.

Public Member Functions

- bool `has` (const `value` &`v`) const
Test if v belongs to this set of values.
- unsigned `index_of` (const `value` &`v`) const
Give the index of value v in this set.
- unsigned `nvalues` () const
Give the number of values in this set.
- `value operator[]` (unsigned `i`) const
Give the i -th value of this set.

10.125.1 Detailed Description

`template<typename E> struct mln::doc::Value_Set< E >`

Documentation class for `mln::Value_Set`.

See also

mln::Value_Set

Definition at line 37 of file doc/value_set.hh.

10.125.2 Member Typedef Documentation**10.125.2.1 `template<typename E> typedef void mln::doc::Value_Set< E >::bkd_viter`**

Backward [Value_Iterator](#) associated type.

Definition at line 49 of file doc/value_set.hh.

10.125.2.2 `template<typename E> typedef void mln::doc::Value_Set< E >::fwd_viter`

Forward [Value_Iterator](#) associated type.

Definition at line 45 of file doc/value_set.hh.

10.125.2.3 `template<typename E> typedef void mln::doc::Value_Set< E >::value`

[Value](#) associated type.

Definition at line 41 of file doc/value_set.hh.

10.125.3 Member Function Documentation**10.125.3.1 `template<typename E> bool mln::doc::Value_Set< E >::has (const value & v) const`**

Test if *v* belongs to this set of values.

Parameters

[in] *v* A value.

Returns

True if *v* is an element of the set of values.

10.125.3.2 `template<typename E> unsigned mln::doc::Value_Set< E >::index_of (const value & v) const`

Give the index of value *v* in this set.

10.125.3.3 `template<typename E> unsigned mln::doc::Value_Set< E >::nvalues () const`

Give the number of values in this set.

10.125.3.4 `template<typename E> value mln::doc::Value_Set< E >::operator[] (unsigned i) const`

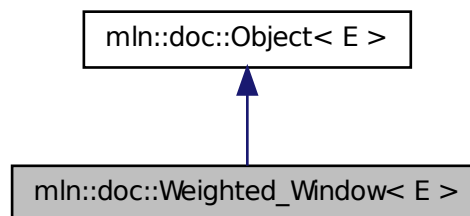
Give the `i`-th value of this set.

10.126 `mln::doc::Weighted_Window< E >` Struct Template Reference

Documentation class for [mln::Weighted_Window](#).

```
#include <weighted_window.hh>
```

Inheritance diagram for `mln::doc::Weighted_Window< E >`:



Public Types

- typedef void [bkd_qiter](#)
[Site_Iterator](#) type associated to this `weighted_window` to browse its points in a backward way.
- typedef void [dpoint](#)
`Dpsite` associated type.
- typedef void [fwd_qiter](#)
[Site_Iterator](#) type associated to this `weighted_window` to browse its points in a forward way.
- typedef void [point](#)
[Site](#) associated type.
- typedef void [weight](#)
`Weight` associated type.
- typedef void [window](#)
[Window](#) associated type.

Public Member Functions

- unsigned [delta](#) () const
Give the maximum coordinate gap between the window center and a window point.
- bool [is_centered](#) () const
Test if the weighted_window is centered.
- bool [is_empty](#) () const
Test if the weighted window is empty.
- E & [sym](#) ()
Apply a central symmetry to the target weighted window.
- const [window](#) & [win](#) () const
Give the corresponding window.

10.126.1 Detailed Description

template<typename E> struct mln::doc::Weighted_Window< E >

Documentation class for [mln::Weighted_Window](#). A weighted_window is the definition of a set of points located around a central point, with a weight associated to each point.

See also

[mln::Weighted_Window](#)

Definition at line 40 of file doc/weighted_window.hh.

10.126.2 Member Typedef Documentation

10.126.2.1 template<typename E > typedef void mln::doc::Weighted_Window< E >::bkd_qiter

[Site_Iterator](#) type associated to this weighted_window to browse its points in a backward way.

Definition at line 51 of file doc/weighted_window.hh.

10.126.2.2 template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint

Dpsite associated type.

Definition at line 57 of file doc/weighted_window.hh.

10.126.2.3 template<typename E > typedef void mln::doc::Weighted_Window< E >::fwd_qiter

[Site_Iterator](#) type associated to this weighted_window to browse its points in a forward way.

Definition at line 46 of file doc/weighted_window.hh.

10.126.2.4 `template<typename E> typedef void mln::doc::Weighted_Window< E>::point`

[Site](#) associated type.

Definition at line 54 of file doc/weighted_window.hh.

10.126.2.5 `template<typename E> typedef void mln::doc::Weighted_Window< E>::weight`

Weight associated type.

Definition at line 60 of file doc/weighted_window.hh.

10.126.2.6 `template<typename E> typedef void mln::doc::Weighted_Window< E>::window`

[Window](#) associated type.

Definition at line 63 of file doc/weighted_window.hh.

10.126.3 Member Function Documentation

10.126.3.1 `template<typename E> unsigned mln::doc::Weighted_Window< E>::delta () const`

Give the maximum coordinate gap between the window center and a window point.

10.126.3.2 `template<typename E> bool mln::doc::Weighted_Window< E>::is_centered () const`

Test if the weighted_window is centered.

A weighted window is centered is the origin belongs to it.

10.126.3.3 `template<typename E> bool mln::doc::Weighted_Window< E>::is_empty () const`

Test if the weighted window is empty.

A weighted_window of null size is empty.

10.126.3.4 `template<typename E> E& mln::doc::Weighted_Window< E>::sym ()`

Apply a central symmetry to the target weighted window.

10.126.3.5 `template<typename E> const window& mln::doc::Weighted_Window< E>::win () const`

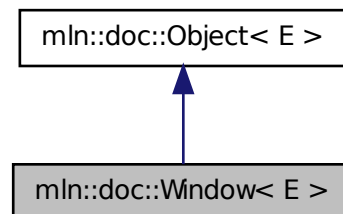
Give the corresponding window.

10.127 `mln::doc::Window< E>` Struct Template Reference

Documentation class for [mln::Window](#).

```
#include <window.hh>
```

Inheritance diagram for mln::doc::Window< E >:



Public Types

- typedef void [bkd_qiter](#)
[Site_Iterator](#) type associated to this window to browse its points in a backward way.
- typedef void [fwd_qiter](#)
[Site_Iterator](#) type associated to this window to browse its points in a forward way.
- typedef void [qiter](#)
[Site_Iterator](#) type associated to this window to browse its points.

10.127.1 Detailed Description

template<typename E> struct mln::doc::Window< E >

Documentation class for [mln::Window](#). A window is the definition of a set of points located around a central point.

See also

[mln::Window](#)

Definition at line 40 of file concept/doc/window.hh.

10.127.2 Member Typedef Documentation

10.127.2.1 template<typename E > typedef void mln::doc::Window< E >::bkd_qiter

[Site_Iterator](#) type associated to this window to browse its points in a backward way.

Definition at line 55 of file concept/doc/window.hh.

10.127.2.2 `template<typename E> typedef void mln::doc::Window< E >::fwd_qiter`

[Site_Iterator](#) type associated to this window to browse its points in a forward way.

Definition at line 50 of file `concept/doc/window.hh`.

10.127.2.3 `template<typename E> typedef void mln::doc::Window< E >::qiter`

[Site_Iterator](#) type associated to this window to browse its points.

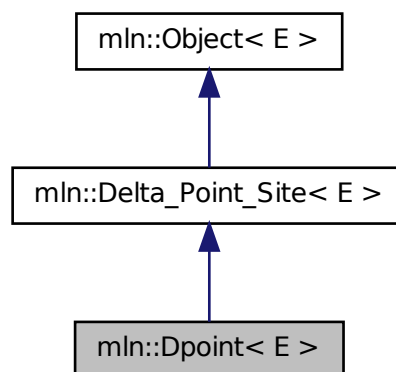
Definition at line 45 of file `concept/doc/window.hh`.

10.128 `mln::Dpoint< E >` Struct Template Reference

Base class for implementation of delta-point classes.

```
#include <dpoint.hh>
```

Inheritance diagram for `mln::Dpoint< E >`:



Public Member Functions

- `const E & to_dpoint () const`
It is a [Dpoint](#) so it returns itself.

10.128.1 Detailed Description

```
template<typename E> struct mln::Dpoint< E >
```

Base class for implementation of delta-point classes. A delta-point is a vector defined by a couple of points.

Given two points, A and B, the vector AB is mapped into the delta-point $D = AB$. Practically one can write:
 $D = B - A$.

See also

[mln::doc::Dpoint](#) for a complete documentation of this class contents.

Definition at line 63 of file concept/dpoint.hh.

10.128.2 Member Function Documentation

10.128.2.1 `template<typename E > const E & mln::Dpoint< E >::to_dpoint () const`
`[inline]`

It is a [Dpoint](#) so it returns itself.

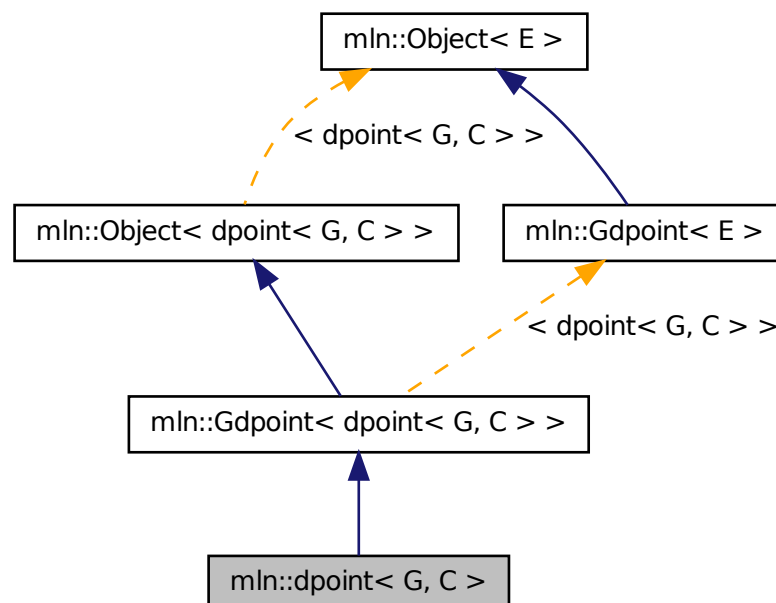
Definition at line 88 of file concept/dpoint.hh.

10.129 mln::dpoint< G, C > Struct Template Reference

Generic delta-point class.

```
#include <dpoint.hh>
```

Inheritance diagram for mln::dpoint< G, C >:



Public Types

- enum { [dim](#) = G::dim }
- typedef C [coord](#)
Coordinate associated type.
- typedef G [grid](#)
Grid associated type.
- typedef [point](#)< G, C > [psite](#)
Psite associated type.
- typedef [point](#)< G, C > [site](#)
Site associated type.
- typedef algebra::vec< G::dim, C > [vec](#)
Algebra vector (vec) associated type.

Public Member Functions

- [dpoint](#) ()
Constructor without argument.
- template<typename C2 >
[dpoint](#) (const algebra::vec< dim, C2 > &v)
Constructor from an algebra vector.
- template<typename F >
[dpoint](#) (const [Function_v2v](#)< F > &f)
Constructor; coordinates are set by function f .
- template<typename Q >
[operator mln::algebra::vec< dpoint< G, C >::dim, Q > \(\)](#) const
Conversion towards a algebra::vec.
- C [operator\[\]](#) (unsigned i) const
Read-only access to the i -th coordinate value.
- C & [operator\[\]](#) (unsigned i)
Read-write access to the i -th coordinate value.
- void [set_all](#) (C c)
Set all coordinates to the value c .
- [vec to_vec](#) () const
Explicit conversion.
- [dpoint](#) (C ind)

- [dpoint](#) (const [literal::zero_t](#) &)
Constructors/assignments with literals.

10.129.1 Detailed Description

template<typename G, typename C> struct mln::dpoint< G, C >

Generic delta-point class. Parameters are G the dimension of the space and C the coordinate type in this space.

Definition at line 58 of file dpoint.hh.

10.129.2 Member Typedef Documentation

10.129.2.1 template<typename G, typename C> typedef C mln::dpoint< G, C >::coord

Coordinate associated type.

Definition at line 76 of file dpoint.hh.

10.129.2.2 template<typename G, typename C> typedef G mln::dpoint< G, C >::grid

Grid associated type.

Definition at line 67 of file dpoint.hh.

10.129.2.3 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite

Psite associated type.

Definition at line 70 of file dpoint.hh.

10.129.2.4 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site

[Site](#) associated type.

Definition at line 73 of file dpoint.hh.

10.129.2.5 template<typename G, typename C> typedef algebra::vec<G::dim, C> mln::dpoint< G, C >::vec

Algebra vector (vec) associated type.

Definition at line 79 of file dpoint.hh.

10.129.3 Member Enumeration Documentation

10.129.3.1 `template<typename G, typename C> anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 64 of file `dpoint.hh`.

10.129.4 Constructor & Destructor Documentation

10.129.4.1 `template<typename G , typename C > mln::dpoint< G, C >::dpoint () [inline]`

Constructor without argument.

Definition at line 152 of file `dpoint.hh`.

10.129.4.2 `template<typename G , typename C > template<typename C2 > mln::dpoint< G, C >::dpoint (const algebra::vec< dim, C2 > & v) [inline]`

Constructor from an algebra vector.

Definition at line 159 of file `dpoint.hh`.

References `mln::dpoint< G, C >::dim`.

10.129.4.3 `template<typename G , typename C> mln::dpoint< G, C >::dpoint (C ind) [inline]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 176 of file `dpoint.hh`.

10.129.4.4 `template<typename G , typename C> mln::dpoint< G, C >::dpoint (const literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 203 of file `dpoint.hh`.

10.129.4.5 `template<typename G , typename C > template<typename F > mln::dpoint< G, C >::dpoint (const Function_v2v< F > & f) [inline]`

Constructor; coordinates are set by function f .

Definition at line 238 of file `dpoint.hh`.

10.129.5 Member Function Documentation

10.129.5.1 `template<typename G , typename C > template<typename Q > mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q > () const [inline]`

Conversion towards a algebra::vec.

Definition at line 257 of file dpoint.hh.

References mln::dpoint< G, C >::to_vec().

10.129.5.2 `template<typename G , typename C > C & mln::dpoint< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

i < dim

Definition at line 144 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

10.129.5.3 `template<typename G , typename C > C mln::dpoint< G, C >::operator[] (unsigned i) const [inline]`

Read-only access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

i < dim

Definition at line 136 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

10.129.5.4 `template<typename G , typename C> void mln::dpoint< G, C >::set_all (C c) [inline]`

Set all coordinates to the value *c*.

Definition at line 248 of file dpoint.hh.

Referenced by mln::win::line< M, i, C >::line().

10.129.5.5 `template<typename G , typename C > dpoint< G, C >::vec mln::dpoint< G, C >::to_vec () const [inline]`

Explicit conversion.

Definition at line 265 of file dpoint.hh.

References `mln::dpoint< G, C >::dim`.

Referenced by `mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >()`.

10.130 `mln::dpoints_bkd_pixter< I >` Class Template Reference

A generic backward iterator on the pixels of a dpoint-based window or neighborhood.

`#include <dpoints_pixter.hh>`

Inherits `Pixel_Iterator< dpoints_bkd_pixter< I > >`, and `pixel_impl< I, dpoints_bkd_pixter< I > >`.

Public Member Functions

- `const I::value & center_val () const`
The value around which this iterator moves.
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (const Generalized_Pixel< Pref > &pxl_ref, const Dps &dps)`
Constructor (using a generalized pixel).
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (I &image, const Dps &dps, const Pref &p_ref)`
Constructor (using an image).
- `void next ()`
Go to the next element.
- `void start ()`
Manipulation.
- `void invalidate ()`
Invalidate the iterator.
- `bool is_valid () const`
Test the iterator validity.
- `void update ()`
Force this iterator to update its location to take into account that its center point may have moved.

10.130.1 Detailed Description

template<typename I> class mln::dpoints_bkd_pixter< I >

A generic backward iterator on the pixels of a dpoint-based window or neighborhood. Parameter *I* is the image type.

Definition at line 140 of file `dpoints_pixter.hh`.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (I & *image*, const Dps & *dps*, const Pref & *p_ref*) [inline]

Constructor (using an image).

Parameters

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p_ref* Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 338 of file `dpoints_pixter.hh`.

10.130.2.2 template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (const Generalized_Pixel< Pref > & *pxl_ref*, const Dps & *dps*) [inline]

Constructor (using a generalized pixel).

Parameters

- [in] *pxl_ref* Center (generalized) pixel to iterate around.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 352 of file `dpoints_pixter.hh`.

10.130.3 Member Function Documentation

10.130.3.1 template<typename I > const I::value & mln::dpoints_bkd_pixter< I >::center_val () const [inline]

The value around which this iterator moves.

Definition at line 367 of file `dpoints_pixter.hh`.

10.130.3.2 template<typename I > void mln::dpoints_bkd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

Definition at line 436 of file `dpoints_pixter.hh`.

10.130.3.3 `template<typename I> bool mln::dpoints_bkd_pixter< I >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 428 of file `dpoints_pixter.hh`.

Referenced by `mln::dpoints_bkd_pixter< I >::update()`.

10.130.3.4 `void mln::Iterator< dpoints_bkd_pixter< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.130.3.5 `template<typename I> void mln::dpoints_bkd_pixter< I >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 409 of file `dpoints_pixter.hh`.

References `mln::dpoints_bkd_pixter< I >::update()`.

10.130.3.6 `template<typename I> void mln::dpoints_bkd_pixter< I >::update () [inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 395 of file `dpoints_pixter.hh`.

References `mln::dpoints_bkd_pixter< I >::is_valid()`.

Referenced by `mln::dpoints_bkd_pixter< I >::start()`.

10.131 `mln::dpoints_fwd_pixter< I >` Class Template Reference

A generic forward iterator on the pixels of a dpoint-based window or neighborhood.

`#include <dpoints_pixter.hh>`

Inherits `Pixel_Iterator< dpoints_fwd_pixter< I > >`, and `pixel_impl< I, dpoints_fwd_pixter< I > >`.

Public Member Functions

- `const I::value & center_val () const`

The value around which this iterator moves.

- template<typename Dps , typename Pref >
[dpoints_fwd_pixter](#) (const [Generalized_Pixel](#)< Pref > &p_xl_ref, const Dps &dps)
Constructor (using a generalized pixel).
- template<typename Dps , typename Pref >
[dpoints_fwd_pixter](#) (I &image, const Dps &dps, const Pref &p_ref)
Constructor (using an image).
- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [update](#) ()
Force this iterator to update its location to take into account that its center point may have moved.

10.131.1 Detailed Description

template<typename I> class mln::dpoints_fwd_pixter< I >

A generic forward iterator on the pixels of a dpoint-based window or neighborhood. Parameter I is the image type.

Definition at line 57 of file dpoints_pixter.hh.

10.131.2 Constructor & Destructor Documentation

**10.131.2.1 template<typename I > template<typename Dps , typename Pref >
mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (I & image, const Dps & dps,
const Pref & p_ref) [inline]**

Constructor (using an image).

Parameters

- [in] **image** The image to iterate over.
- [in] **dps** An object (neighborhood or window) that can provide a set of delta-points.
- [in] **p_ref** Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 224 of file dpoints_pixter.hh.

10.131.2.2 `template<typename I > template<typename Dps , typename Pref >
mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (const Generalized_Pixel< Pref
> & pxl_ref, const Dps & dps) [inline]`

Constructor (using a generalized pixel).

Parameters

[in] *pxl_ref* Center (generalized) pixel to iterate around.

[in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 241 of file `dpoints_pixter.hh`.

10.131.3 Member Function Documentation

10.131.3.1 `template<typename I > const I::value & mln::dpoints_fwd_pixter< I >::center_val () const [inline]`

The value around which this iterator moves.

Definition at line 256 of file `dpoints_pixter.hh`.

10.131.3.2 `template<typename I > void mln::dpoints_fwd_pixter< I >::invalidate () [inline]`

Invalidate the iterator.

Definition at line 325 of file `dpoints_pixter.hh`.

10.131.3.3 `template<typename I > bool mln::dpoints_fwd_pixter< I >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 317 of file `dpoints_pixter.hh`.

Referenced by `mln::dpoints_fwd_pixter< I >::update()`.

10.131.3.4 `void mln::Iterator< dpoints_fwd_pixter< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.131.3.5 template<typename I > void mln::dpoints_fwd_piter< I >::start () [inline]

Manipulation.

Start an iteration.

Definition at line 298 of file dpoints_piter.hh.

References mln::dpoints_fwd_piter< I >::update().

10.131.3.6 template<typename I > void mln::dpoints_fwd_piter< I >::update () [inline]

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 284 of file dpoints_piter.hh.

References mln::dpoints_fwd_piter< I >::is_valid().

Referenced by mln::dpoints_fwd_piter< I >::start().

10.132 mln::dpsites_bkd_piter< V > Class Template Reference

A generic backward iterator on points of windows and of neighborhoods.

```
#include <dpsites_piter.hh>
```

Inherits site_relative_iterator_base< V, dpsites_bkd_piter< V > >.

Public Member Functions

- template<typename P >
 [dpsites_bkd_piter](#) (const V &v, const P &c)

Constructor.

- [dpsites_bkd_piter](#) ()

Constructor without argument.

- void [next](#) ()

Go to the next element.

10.132.1 Detailed Description

```
template<typename V> class mln::dpsites_bkd_piter< V >
```

A generic backward iterator on points of windows and of neighborhoods. The parameter V is the type of std::vector enclosing structure.

Definition at line 94 of file dpsites_piter.hh.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 `template<typename V > template<typename P > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter (const V & v, const P & c) [inline]`

Constructor.

Parameters

- [in] `v` [Object](#) that can provide an array of delta-points.
- [in] `c` Center point to iterate around.

Definition at line 217 of file `dpsites_piter.hh`.

10.132.2.2 `template<typename V > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter () [inline]`

Constructor without argument.

Definition at line 210 of file `dpsites_piter.hh`.

10.132.3 Member Function Documentation

10.132.3.1 `void mln::Site_Iterator< dpsites_bkd_piter< V > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.133 mln::dpsites_fwd_piter< V > Class Template Reference

A generic forward iterator on points of windows and of neighborhoods.

```
#include <dpsites_piter.hh>
```

Inherits `site_relative_iterator_base< V, dpsites_fwd_piter< V > >`.

Public Member Functions

- `template<typename P > dpsites_fwd_piter (const V &v, const P &c)`
Constructor.
- `dpsites_fwd_piter ()`
Constructor without argument.

- void [next](#) ()

Go to the next element.

10.133.1 Detailed Description

template<typename V> class mln::dpsites_fwd_piter< V >

A generic forward iterator on points of windows and of neighborhoods. The parameter V is the type of std::vector enclosing structure.

Definition at line 48 of file dpsites_piter.hh.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 template<typename V > template<typename P > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter (const V & v, const P & c) [inline]

Constructor.

Parameters

[in] **v** [Object](#) that can provide an array of delta-points.

[in] **c** Center point to iterate around.

Definition at line 149 of file dpsites_piter.hh.

10.133.2.2 template<typename V > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter () [inline]

Constructor without argument.

Definition at line 142 of file dpsites_piter.hh.

10.133.3 Member Function Documentation

10.133.3.1 void mln::Site_Iterator< dpsites_fwd_piter< V > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.134 mln::Edge< E > Struct Template Reference

edge category flag type.

```
#include <edge.hh>
```

10.134.1 Detailed Description

template<typename E> struct mln::Edge< E >

edge category flag type.

Definition at line 52 of file edge.hh.

10.135 mln::edge_image< P, V, G > Class Template Reference

[Image](#) based on graph edges.

```
#include <edge_image.hh>
```

Inherits [image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t>, edge_image< P, V, G >>](#).

Public Types

- typedef [graph_elt_neighborhood< G, p_edges< G, site_function_t >> edge_nbh_t](#)
Neighborhood type.
- typedef [graph_elt_window< G, p_edges< G, site_function_t >> edge_win_t](#)
Edge Window type.
- typedef G [graph_t](#)
The type of the underlying graph.
- typedef [edge_nbh_t nbh_t](#)
Default Neighborhood type.
- typedef internal::efsite_selector< P, G >::[site_function_t site_function_t](#)
Function mapping graph elements to sites.
- typedef [edge_image< tag::psite_< P >, tag::value_< V >, tag::graph_< G >> skeleton](#)
Skeleton type.
- typedef [edge_win_t win_t](#)
Default Window type.

Public Member Functions

- [edge_image](#) ()
Constructors.
- rvalue [operator](#)() (unsigned e_id) const
Value accessors/operators overloads.

10.135.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::edge_image< P, V, G >`

[Image](#) based on graph edges.

Definition at line 123 of file core/image/edge_image.hh.

10.135.2 Member Typedef Documentation

10.135.2.1 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_nbh_t`

[Neighborhood](#) type.

Definition at line 153 of file core/image/edge_image.hh.

10.135.2.2 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_win_t`

[Edge Window](#) type.

Definition at line 151 of file core/image/edge_image.hh.

10.135.2.3 `template<typename P, typename V, typename G = util::graph> typedef G mln::edge_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 138 of file core/image/edge_image.hh.

10.135.2.4 `template<typename P, typename V, typename G = util::graph> typedef edge_nbh_t mln::edge_image< P, V, G >::nbh_t`

Default [Neighborhood](#) type.

Definition at line 159 of file core/image/edge_image.hh.

10.135.2.5 `template<typename P, typename V, typename G = util::graph> typedef
internal::efsite_selector<P,G>::site_function_t mln::edge_image< P, V, G
>::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 147 of file core/image/edge_image.hh.

10.135.2.6 `template<typename P, typename V, typename G = util::graph> typedef edge_image<
tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::edge_image< P, V, G
>::skeleton`

Skeleton type.

Definition at line 143 of file core/image/edge_image.hh.

10.135.2.7 `template<typename P, typename V, typename G = util::graph> typedef edge_win_t
mln::edge_image< P, V, G >::win_t`

Default [Window](#) type.

Definition at line 156 of file core/image/edge_image.hh.

10.135.3 Constructor & Destructor Documentation

10.135.3.1 `template<typename P , typename V , typename G > mln::edge_image< P, V, G
>::edge_image() [inline]`

Constructors.

Definition at line 248 of file core/image/edge_image.hh.

10.135.4 Member Function Documentation

10.135.4.1 `template<typename P , typename V , typename G > edge_image< P, V, G >::rvalue
mln::edge_image< P, V, G >::operator() (unsigned e_id) const`

[Value](#) accessors/operators overloads.

Definition at line 302 of file core/image/edge_image.hh.

10.136 mln::extended< I > Struct Template Reference

Makes an image become restricted by a point set.

`#include <extended.hh>`

Inherits `image_domain_morpher< I, box< I::site >, extended< I > >`.

Public Types

- `typedef tag::image_< I > skeleton`

Skeleton.

- typedef I::value [value](#)
Value type.

Public Member Functions

- const [box](#)< typename I::site > & [domain](#) () const
Give the definition domain.
- [extended](#) ()
Constructor without argument.
- [extended](#) (I &ima, const [box](#)< typename I::site > &b)
Constructor.

10.136.1 Detailed Description

template<typename I> struct mln::extended< I >

Makes an image become restricted by a point set.

Definition at line 92 of file extended.hh.

10.136.2 Member Typedef Documentation

10.136.2.1 template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton

Skeleton.

Definition at line 102 of file extended.hh.

10.136.2.2 template<typename I> typedef I ::value mln::extended< I >::value

Value type.

Definition at line 99 of file extended.hh.

10.136.3 Constructor & Destructor Documentation

10.136.3.1 template<typename I > mln::extended< I >::extended () [inline]

Constructor without argument.

Definition at line 169 of file extended.hh.

10.136.3.2 `template<typename I> mln::extended< I >::extended (I & ima, const box< typename I::site > & b) [inline]`

Constructor.

Definition at line 175 of file extended.hh.

10.136.4 Member Function Documentation

10.136.4.1 `template<typename I> const box< typename I::site > & mln::extended< I >::domain () const [inline]`

Give the definition domain.

Definition at line 192 of file extended.hh.

10.137 mln::extension_fun< I, F > Class Template Reference

Extends the domain of an image with a function.

`#include <extension_fun.hh>`

Inherits `image_identity< I, I::domain_t, extension_fun< I, F > >`.

Public Types

- `typedef I::value rvalue`
Return type of read-only access.
- `typedef extension_fun< tag::image_< I >, tag::function_< F > > skeleton`
Skeleton.
- `typedef I::value value`
Image value type.

Public Member Functions

- `const F & extension () const`
Give the extension function.
- `extension_fun (I &ima, const F &fun)`
*Constructor from an image *ima* and a function *fun*.*
- `extension_fun ()`
Constructor without argument.
- `template<typename P>
bool has (const P &p) const`
*Test if *p* is valid.*

- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)

Read-write access to the image value located at site p.

- I::value [operator\(\)](#) (const typename I::psite &p) const

Read-only access to the image value located at site p.

10.137.1 Detailed Description

template<typename I, typename F> class mln::extension_fun< I, F >

Extends the domain of an image with a function.

Definition at line 99 of file extension_fun.hh.

10.137.2 Member Typedef Documentation

10.137.2.1 template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::rvalue

Return type of read-only access.

Definition at line 112 of file extension_fun.hh.

10.137.2.2 template<typename I, typename F> typedef extension_fun< tag::image_<I>, tag::function_<F> > mln::extension_fun< I, F >::skeleton

Skeleton.

Definition at line 106 of file extension_fun.hh.

10.137.2.3 template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::value

[Image](#) value type.

Definition at line 109 of file extension_fun.hh.

10.137.3 Constructor & Destructor Documentation

**10.137.3.1 template<typename I, typename F > mln::extension_fun< I, F >::extension_fun ()
[inline]**

Constructor without argument.

Definition at line 178 of file extension_fun.hh.

10.137.3.2 `template<typename I, typename F> mln::extension_fun< I, F >::extension_fun (I & ima, const F & fun) [inline]`

Constructor from an image `ima` and a function `fun`.

Definition at line 184 of file `extension_fun.hh`.

10.137.4 Member Function Documentation

10.137.4.1 `template<typename I, typename F> const F & mln::extension_fun< I, F >::extension () const [inline]`

Give the extension function.

Definition at line 243 of file `extension_fun.hh`.

10.137.4.2 `template<typename I, typename F> template<typename P> bool mln::extension_fun< I, F >::has (const P & p) const [inline]`

Test if `p` is valid.

It returns always true, assuming that the function is valid for any `p`.

Definition at line 201 of file `extension_fun.hh`.

10.137.4.3 `template<typename I, typename F> internal::morpher_lvalue_< I >::ret mln::extension_fun< I, F >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file `extension_fun.hh`.

10.137.4.4 `template<typename I, typename F> I::value mln::extension_fun< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 209 of file `extension_fun.hh`.

10.138 mln::extension_ima< I, J > Class Template Reference

Extends the domain of an image with an image.

`#include <extension_ima.hh>`

Inherits `image_identity< I, I::domain_t, extension_ima< I, J > >`.

Public Types

- typedef `I::value` [rvalue](#)

Return type of read-only access.

- typedef [extension_ima](#)< tag::image_< I >, tag::ext_< J > > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Image value type.

Public Member Functions

- const J & [extension](#) () const
Read-only access to the extension domain (image).
- [extension_ima](#) (I &ima, const J &ext)
*Constructor from an image *ima* and a function *ext*.*
- [extension_ima](#) ()
Constructor without argument.
- template<typename P >
bool [has](#) (const P &p) const
*Test if *p* is valid.*
- internal::morpher_lvalue_< I >::ret [operator](#)() (const typename I::psite &p)
*Read-write access to the image value located at site *p*.*
- I::value [operator](#)() (const typename I::psite &p) const
*Read-only access to the image value located at site *p*.*

10.138.1 Detailed Description

template<typename I, typename J> class mln::extension_ima< I, J >

Extends the domain of an image with an image.

Definition at line 97 of file extension_ima.hh.

10.138.2 Member Typedef Documentation

10.138.2.1 template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::rvalue

Return type of read-only access.

Definition at line 111 of file extension_ima.hh.

10.138.2.2 `template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton`

Skeleton.

Definition at line 105 of file extension_ima.hh.

10.138.2.3 `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value`

[Image](#) value type.

Definition at line 108 of file extension_ima.hh.

10.138.3 Constructor & Destructor Documentation

10.138.3.1 `template<typename I, typename J> mln::extension_ima< I, J >::extension_ima () [inline]`

Constructor without argument.

Definition at line 173 of file extension_ima.hh.

10.138.3.2 `template<typename I, typename J> mln::extension_ima< I, J >::extension_ima (I & ima, const J & ext) [inline]`

Constructor from an image `ima` and a function `ext`.

Definition at line 179 of file extension_ima.hh.

10.138.4 Member Function Documentation

10.138.4.1 `template<typename I, typename J> const J & mln::extension_ima< I, J >::extension () const [inline]`

Read-only access to the extension domain (image).

Definition at line 244 of file extension_ima.hh.

10.138.4.2 `template<typename I, typename J> template<typename P> bool mln::extension_ima< I, J >::has (const P & p) const [inline]`

Test if `p` is valid.

Definition at line 196 of file extension_ima.hh.

10.138.4.3 `template<typename I, typename J> internal::morpher_lvalue_< I >::ret mln::extension_ima< I, J >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file extension_ima.hh.

10.138.4.4 `template<typename I , typename J > I::value mln::extension_ima< I, J >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site *p*;

Definition at line 208 of file extension_ima.hh.

10.139 mln::extension_val< I > Class Template Reference

Extends the domain of an image with a value.

```
#include <extension_val.hh>
```

Inherits image_identity< I, I::domain_t, extension_val< I > >.

Public Types

- typedef I::value [rvalue](#)
Return type of read-only access.
- typedef [extension_val](#)< tag::image_< I > > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Image value type.

Public Member Functions

- void [change_extension](#) (const typename I::value &val)
Change the value of the extension domain.
- const I::value & [extension](#) () const
Read-only access to the value of the extension domain.
- [extension_val](#) (I &ima, const typename I::value &val)
*Constructor from an image *ima* and a value *val*.*
- [extension_val](#) ()
Constructor without argument.
- template<typename P >
bool [has](#) (const P &p) const
*Test if *p* is valid. It returns always true.*
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)
*Read-write access to the image value located at site *p*.*
- I::value [operator\(\)](#) (const typename I::psite &p) const
*Read-only access to the image value located at site *p*;*

10.139.1 Detailed Description

template<typename I> class mln::extension_val< I >

Extends the domain of an image with a value.

Definition at line 98 of file extension_val.hh.

10.139.2 Member Typedef Documentation

10.139.2.1 template<typename I> typedef I::value mln::extension_val< I >::rvalue

Return type of read-only access.

Definition at line 110 of file extension_val.hh.

**10.139.2.2 template<typename I> typedef extension_val< tag::image_<I> >
mln::extension_val< I >::skeleton**

Skeleton.

Definition at line 104 of file extension_val.hh.

10.139.2.3 template<typename I> typedef I::value mln::extension_val< I >::value

[Image](#) value type.

Definition at line 107 of file extension_val.hh.

10.139.3 Constructor & Destructor Documentation

10.139.3.1 template<typename I> mln::extension_val< I >::extension_val () [inline]

Constructor without argument.

Definition at line 176 of file extension_val.hh.

**10.139.3.2 template<typename I> mln::extension_val< I >::extension_val (I & *ima*, const
typename I::value & *val*) [inline]**

Constructor from an image *ima* and a value *val*.

Definition at line 182 of file extension_val.hh.

10.139.4 Member Function Documentation

**10.139.4.1 template<typename I> void mln::extension_val< I >::change_extension (const
typename I::value & *val*) [inline]**

Change the value of the extension domain.

Definition at line 247 of file extension_val.hh.

10.139.4.2 `template<typename I> const I::value & mln::extension_val< I >::extension () const [inline]`

Read-only access to the value of the extension domain.

Definition at line 238 of file extension_val.hh.

10.139.4.3 `template<typename I> template<typename P> bool mln::extension_val< I >::has (const P & p) const [inline]`

Test if *p* is valid. It returns always true.

Definition at line 199 of file extension_val.hh.

10.139.4.4 `template<typename I> internal::morpher_lvalue_< I >::ret mln::extension_val< I >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site *p*.

Definition at line 220 of file extension_val.hh.

10.139.4.5 `template<typename I> I::value mln::extension_val< I >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site *p*;

Definition at line 207 of file extension_val.hh.

10.140 mln::flat_image< T, S > Struct Template Reference

[Image](#) with a single value.

```
#include <flat_image.hh>
```

Inherits `image_primary< T, S, flat_image< T, S > >`.

Public Types

- `typedef T & lvalue`
Return type of read-write access.
- `typedef const T & rvalue`
Return type of read-only access.
- `typedef flat_image< tag::value_< T >, tag::domain_< S > > skeleton`
Skeleton.
- `typedef T value`
Value associated type.

Public Member Functions

- `const S & domain () const`
Give the definition domain.
- `flat_image (const T &val, const S &pset)`
Constructor.
- `flat_image ()`
Constructor without argument.
- `bool has (const typename S::psite &p) const`
Test if p is valid: always return true.
- `const T & operator() (const typename S::psite &p) const`
Read-only access to the image value located at point p .
- `T & operator() (const typename S::psite &p)`
Read-write access to the image value located at point p .

10.140.1 Detailed Description

`template<typename T, typename S> struct mln::flat_image< T, S >`

[Image](#) with a single value.

Definition at line 105 of file `flat_image.hh`.

10.140.2 Member Typedef Documentation

10.140.2.1 `template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue`

Return type of read-write access.

Definition at line 118 of file `flat_image.hh`.

10.140.2.2 `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

Definition at line 115 of file `flat_image.hh`.

10.140.2.3 `template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton`

Skeleton.

Definition at line 108 of file `flat_image.hh`.

10.140.2.4 `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

Value associated type.

Definition at line 112 of file flat_image.hh.

10.140.3 Constructor & Destructor Documentation

10.140.3.1 `template<typename T , typename S > mln::flat_image< T, S >::flat_image ()` `[inline]`

Constructor without argument.

Definition at line 191 of file flat_image.hh.

10.140.3.2 `template<typename T , typename S > mln::flat_image< T, S >::flat_image (const T` `& val, const S & pset) [inline]`

Constructor.

Definition at line 197 of file flat_image.hh.

10.140.4 Member Function Documentation

10.140.4.1 `template<typename T , typename S > const S & mln::flat_image< T, S >::domain ()` `const [inline]`

Give the definition domain.

Definition at line 214 of file flat_image.hh.

10.140.4.2 `template<typename T , typename S > bool mln::flat_image< T, S >::has (const` `typename S::psite & p) const [inline]`

Test if p is valid: always return true.

Definition at line 222 of file flat_image.hh.

10.140.4.3 `template<typename T , typename S > const T & mln::flat_image< T, S >::operator() (` `const typename S::psite & p) const [inline]`

Read-only access to the image value located at point p.

Definition at line 230 of file flat_image.hh.

10.140.4.4 `template<typename T , typename S > T & mln::flat_image< T, S >::operator() (` `const typename S::psite & p) [inline]`

Read-write access to the image value located at point p.

Definition at line 239 of file flat_image.hh.

10.141 mln::fun::from_accu< A > Struct Template Reference

Wrap an accumulator into a function.

```
#include <from_accu.hh>
```

Inherits mln::fun::unary_param< from_accu< A >, A * >.

10.141.1 Detailed Description

template<typename A> struct mln::fun::from_accu< A >

Wrap an accumulator into a function.

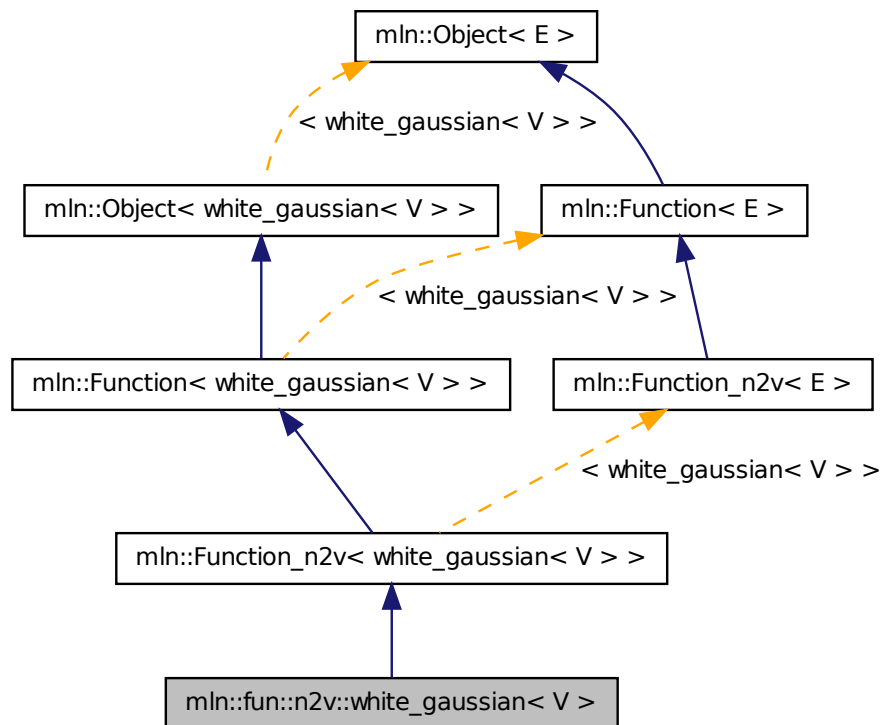
Definition at line 49 of file from_accu.hh.

10.142 mln::fun::n2v::white_gaussian< V > Struct Template Reference

Generate a White Gaussian Noise.

```
#include <white_gaussian.hh>
```

Inheritance diagram for mln::fun::n2v::white_gaussian< V >:



10.142.1 Detailed Description

```
template<typename V> struct mln::fun::n2v::white_gaussian< V >
```

Generate a White Gaussian Noise. Reference: <http://www.dspguru.com/dsp/howtos/how-to-generate-white-gaussian-noise>

Definition at line 56 of file `white_gaussian.hh`.

10.143 mln::fun::p2b::antilogy Struct Reference

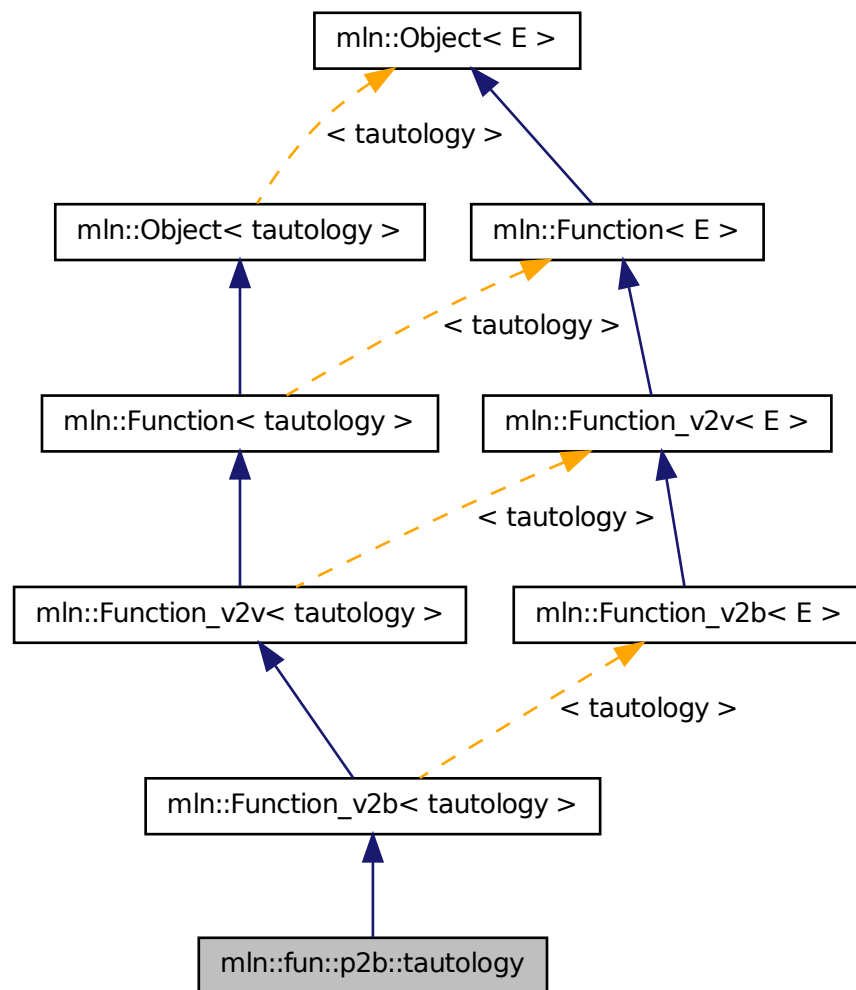
A `p2b` function always returning `false`.

```
#include <antilogy.hh>
```



```
#include <tautology.hh>
```

Inheritance diagram for mln::fun::p2b::tautology:



10.144.1 Detailed Description

A [p2b](#) function always returning `true`. A simpler name would be 'true', but this is not a valid C++ identifier, as `true` is a keyword of the language.

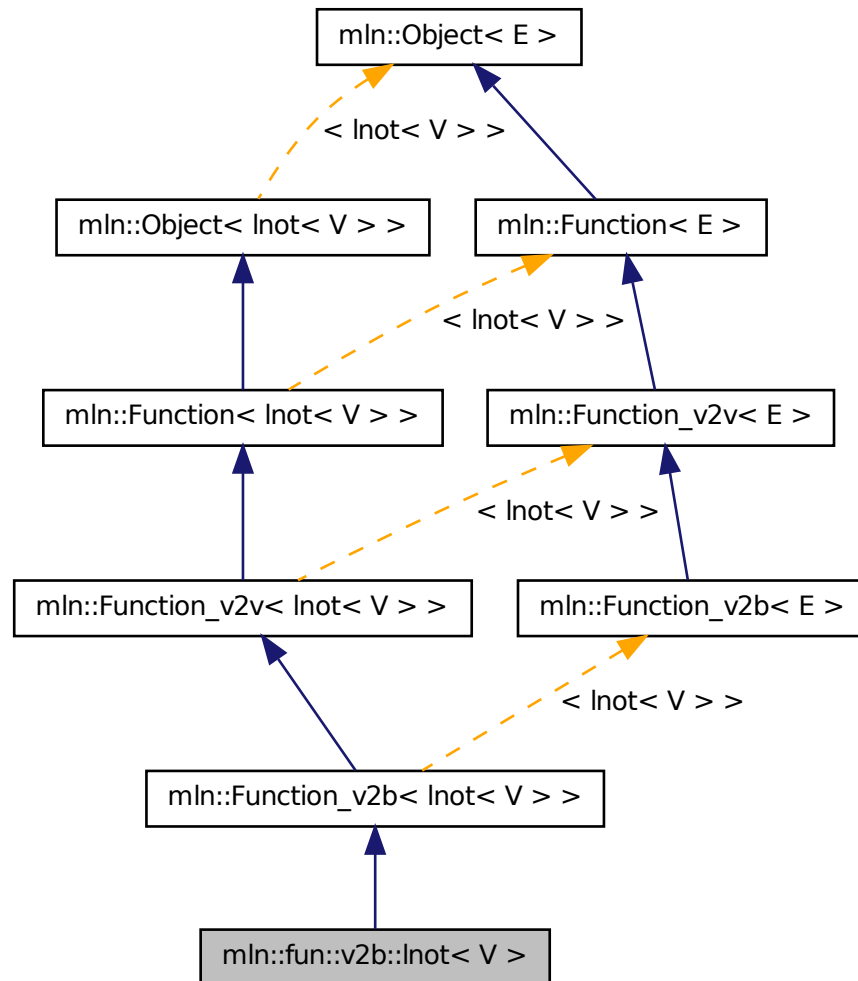
Definition at line 50 of file `tautology.hh`.

10.145 mln::fun::v2b::lnot< V > Struct Template Reference

Functor computing logical-not on a value.

```
#include <lnot.hh>
```

Inheritance diagram for mln::fun::v2b::lnot< V >:



10.145.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::lnot< V >
```

Functor computing logical-not on a value.

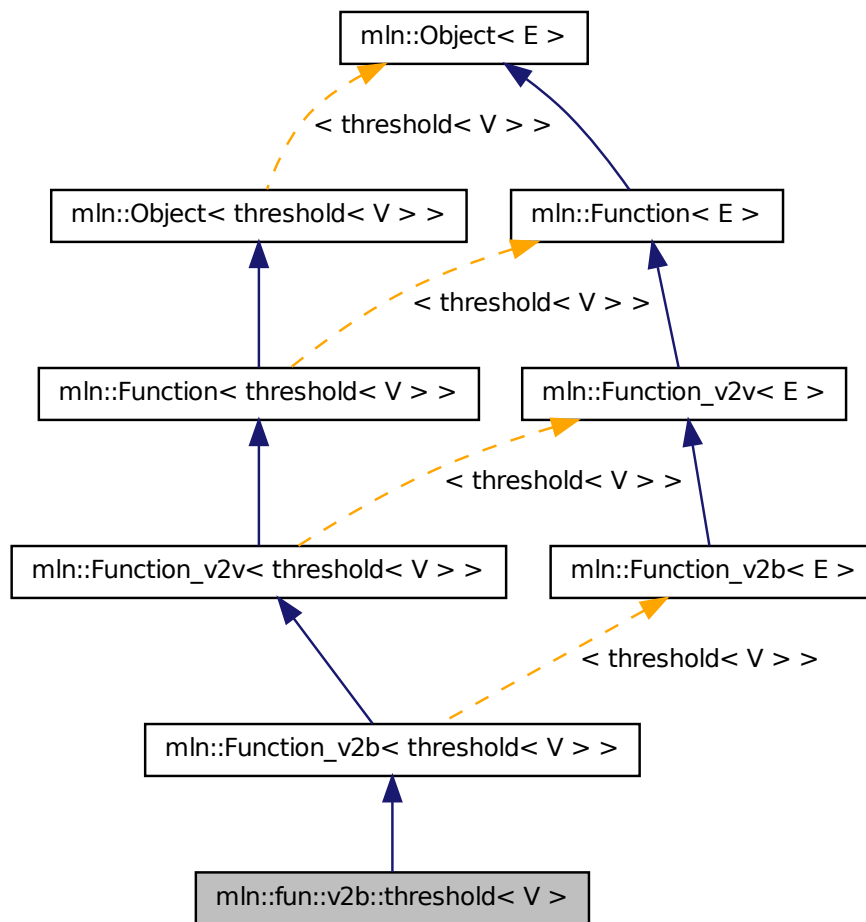
Definition at line 47 of file `lnot.hh`.

10.146 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



10.146.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::threshold< V >
```

Threshold function. $f(v) = (v \geq \text{threshold})$.

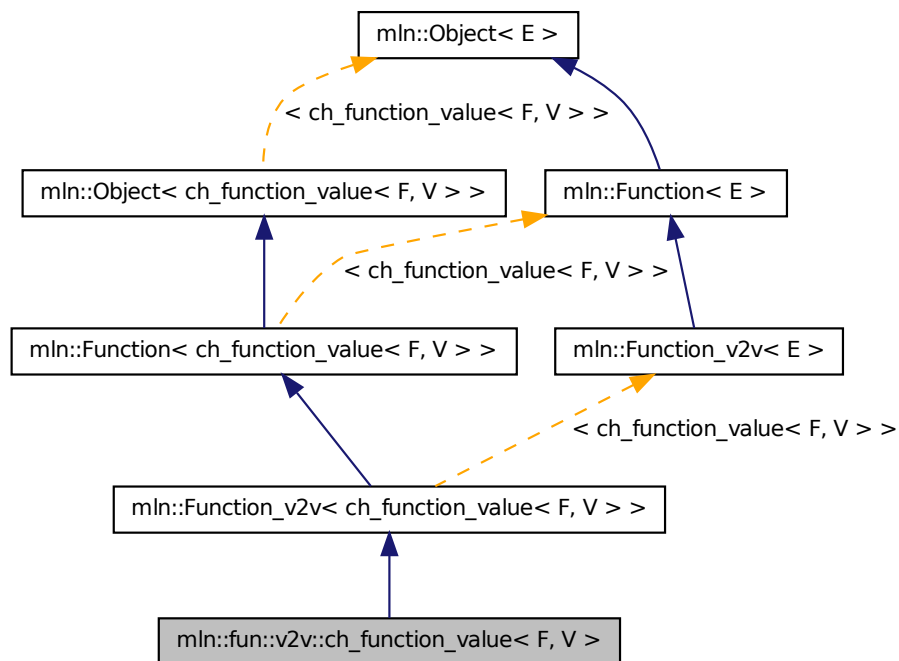
Definition at line 48 of file fun/v2b/threshold.hh.

10.147 mln::fun::v2v::ch_function_value< F, V > Class Template Reference

Wrap a function [v2v](#) and convert its result to another type.

```
#include <ch_function_value.hh>
```

Inheritance diagram for mln::fun::v2v::ch_function_value< F, V >:



10.147.1 Detailed Description

```
template<typename F, typename V> class mln::fun::v2v::ch_function_value< F, V >
```

Wrap a function [v2v](#) and convert its result to another type.

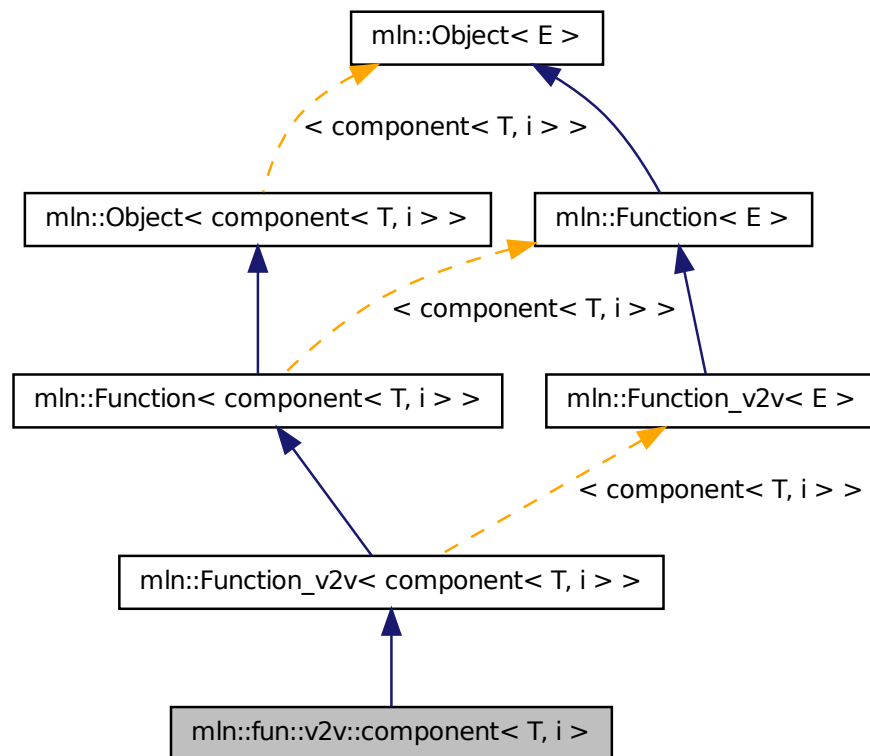
Definition at line 52 of file `fun/v2v/ch_function_value.hh`.

10.148 mln::fun::v2v::component< T, i > Struct Template Reference

Functor that accesses the *i*-th component of a value.

```
#include <component.hh>
```


Inheritance diagram for mln::fun::v2v::component< T, i >:



10.148.1 Detailed Description

```
template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >
```

Functor that accesses the i-th component of a value.

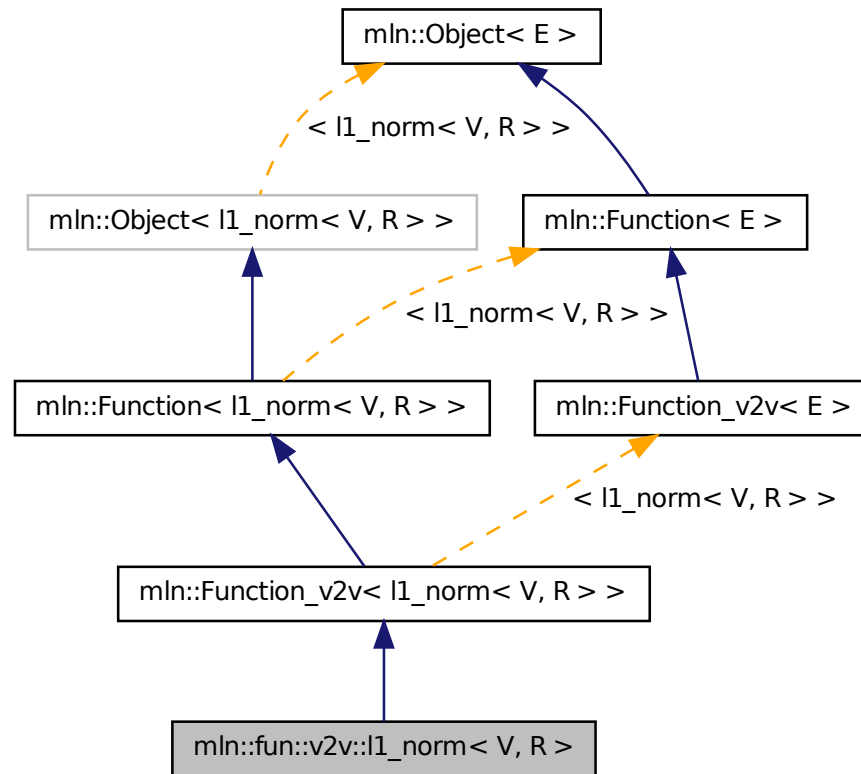
Definition at line 51 of file component.hh.

10.149 mln::fun::v2v::l1_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::l1_norm< V, R >`:



10.149.1 Detailed Description

template<typename V, typename R> struct mln::fun::v2v::l1_norm< V, R >

L1-norm. V is the type of input values; R is the result type.

See also

[mln::norm::l1](#).

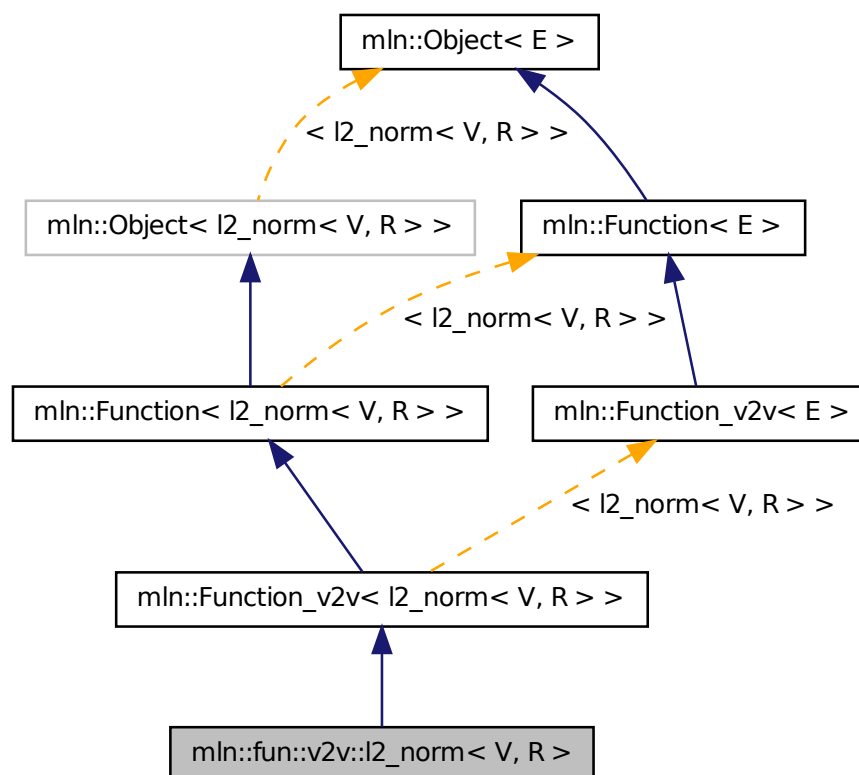
Definition at line 57 of file `v2v/norm.hh`.

10.150 mln::fun::v2v::l2_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2v::l2_norm< V, R >:



10.150.1 Detailed Description

template<typename V, typename R> struct mln::fun::v2v::l2_norm< V, R >

L2-norm. V is the type of input values; R is the result type.

See also

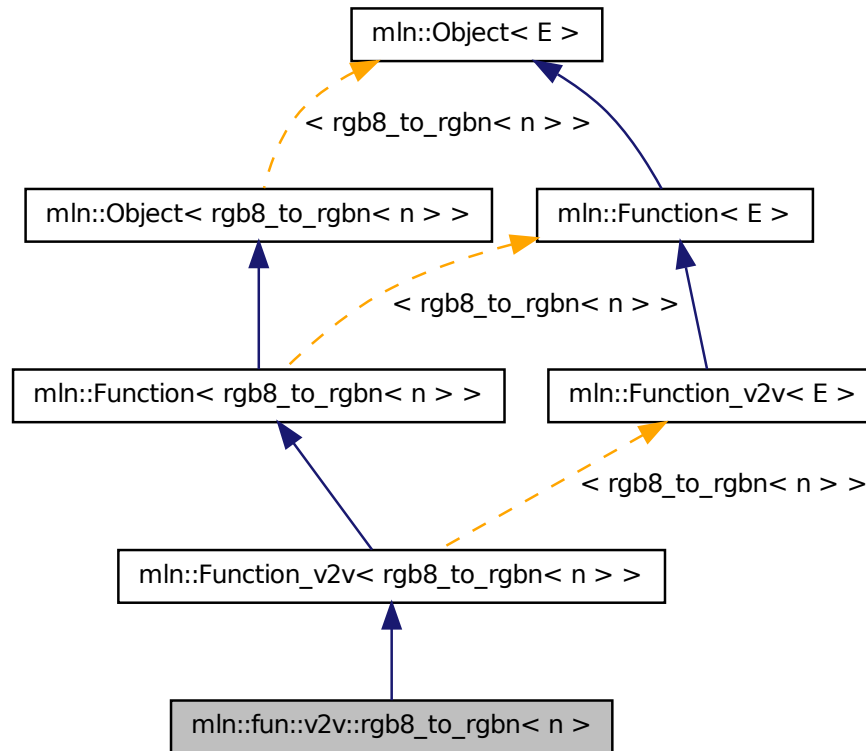
`mln::norm::l2`.

Definition at line 69 of file `v2v/norm.hh`.

10.151 mln::fun::v2v::linear< V, T, R > Struct Template Reference

Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.

Inheritance diagram for `mln::fun::v2v::rgb8_to_rgn< n >`:



Public Member Functions

- `result operator()` (const `argument` &c) const

Convert a rgb8 value to a rgn, $n < 8$.

10.153.1 Detailed Description

`template<unsigned n> struct mln::fun::v2v::rgb8_to_rgn< n >`

Convert a rgb8 value to a rgn, $n < 8$.

Parameters

n defines the output quantification used for the transformation.

Definition at line 56 of file `rgb8_to_rgn.hh`.

10.153.2 Member Function Documentation

10.153.2.1 `template<unsigned n> rgb8_to_rgbn< n >::result mln::fun::v2v::rgb8_to_rgbn< n >::operator() (const argument & c) const`

Convert a rgb8 value to a rgn, $n < 8$.

Parameters

[in] *v* the rgb8 value to convert.

Conversion is done by computing the size by which we divide each rgb component.

Parameters

n defines the output quantification used for the transformation.

Definition at line 83 of file rgb8_to_rgbn.hh.

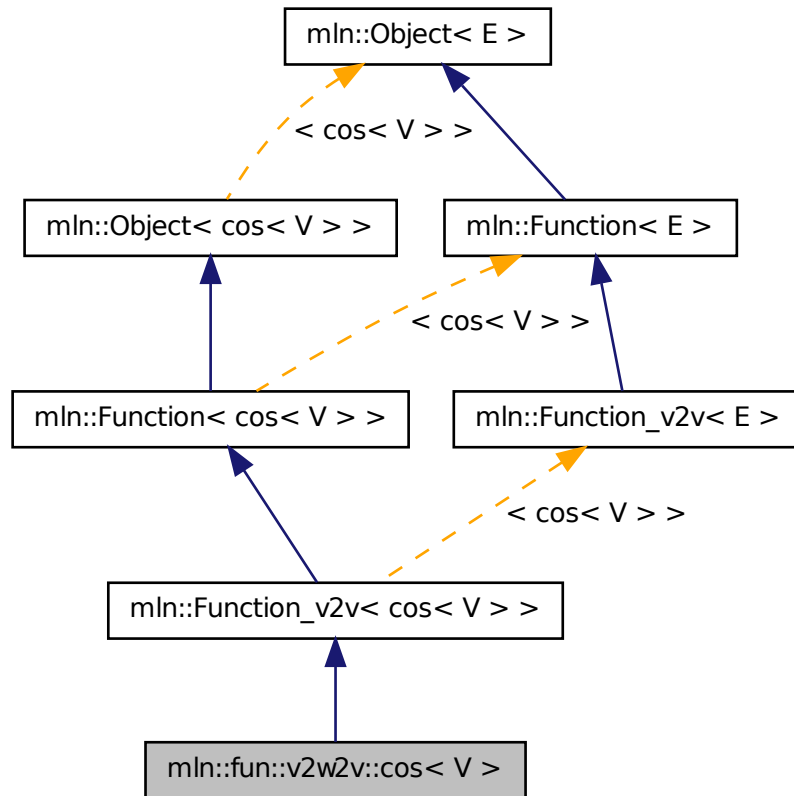
References mln::value::rgb< n >::red().

10.154 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```

Inheritance diagram for `mln::fun::v2w2v::cos< V >`:



10.154.1 Detailed Description

`template<typename V> struct mln::fun::v2w2v::cos< V >`

Cosinus bijective functor. `V` is the type of input values and the result type.

See also

`mln::math::cos`.

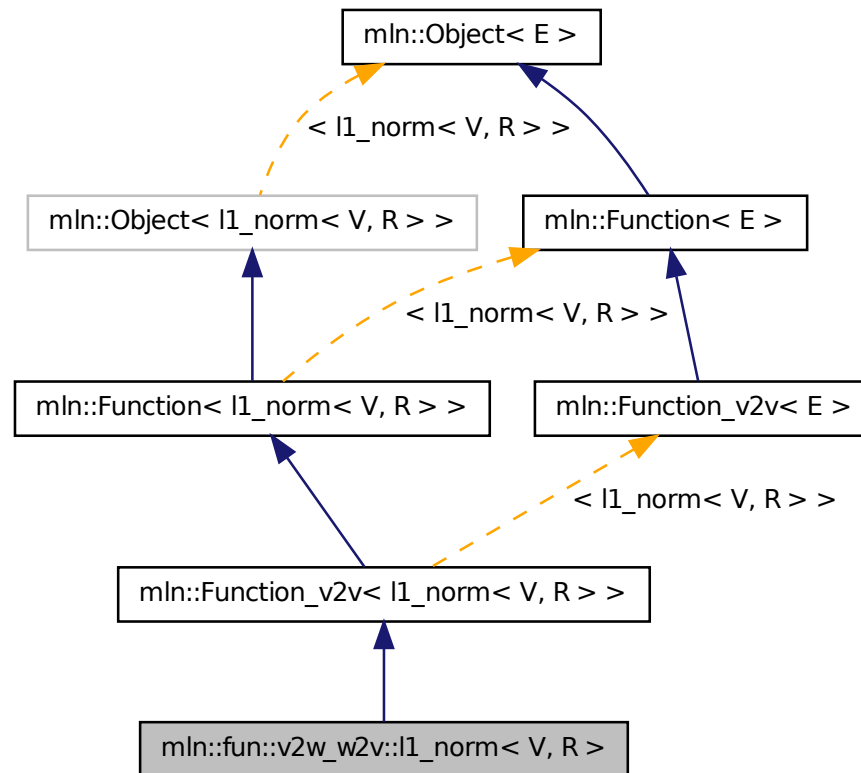
Definition at line 59 of file `fun/v2w2v/cos.hh`.

10.155 `mln::fun::v2w_w2v::l1_norm< V, R >` Struct Template Reference

L1-norm.


```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w_w2v::l1_norm< V, R >:



10.155.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l1_norm< V, R >
```

L1-norm. `V` is the type of input values; `R` is the result type.

See also

[mln::norm::l1](#).

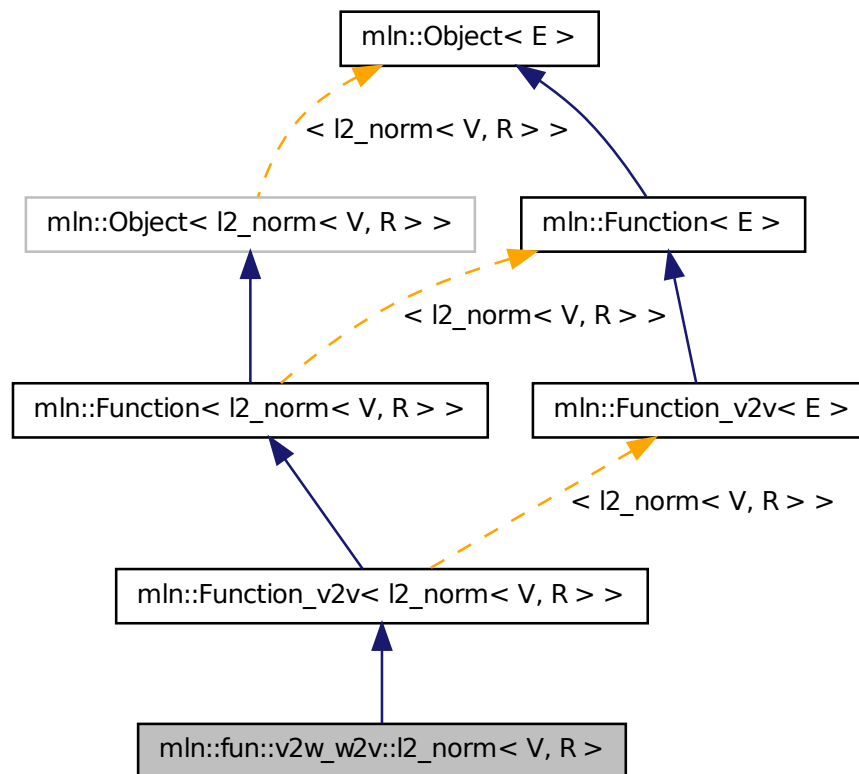
Definition at line 56 of file `v2w_w2v/norm.hh`.

10.156 mln::fun::v2w_w2v::l2_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w_w2v::l2_norm< V, R >:



10.156.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l2_norm< V, R >
```

L2-norm. V is the type of input values; R is the result type.

See also

`mln::norm::l2`.

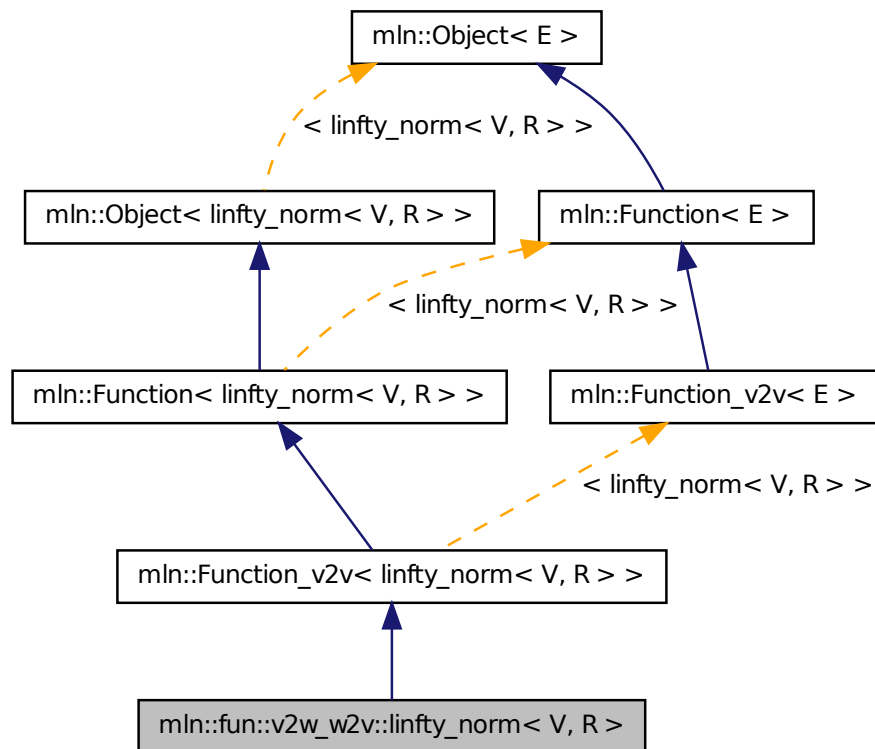
Definition at line 69 of file `v2w_w2v/norm.hh`.

10.157 mln::fun::v2w_w2v::linfty_norm< V, R > Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w_w2v::linfty_norm< V, R >:



10.157.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::linfty_norm< V, R >
```

L-infty norm. V is the type of input values; R is the result type.

See also

[mln::norm::linfty](#).

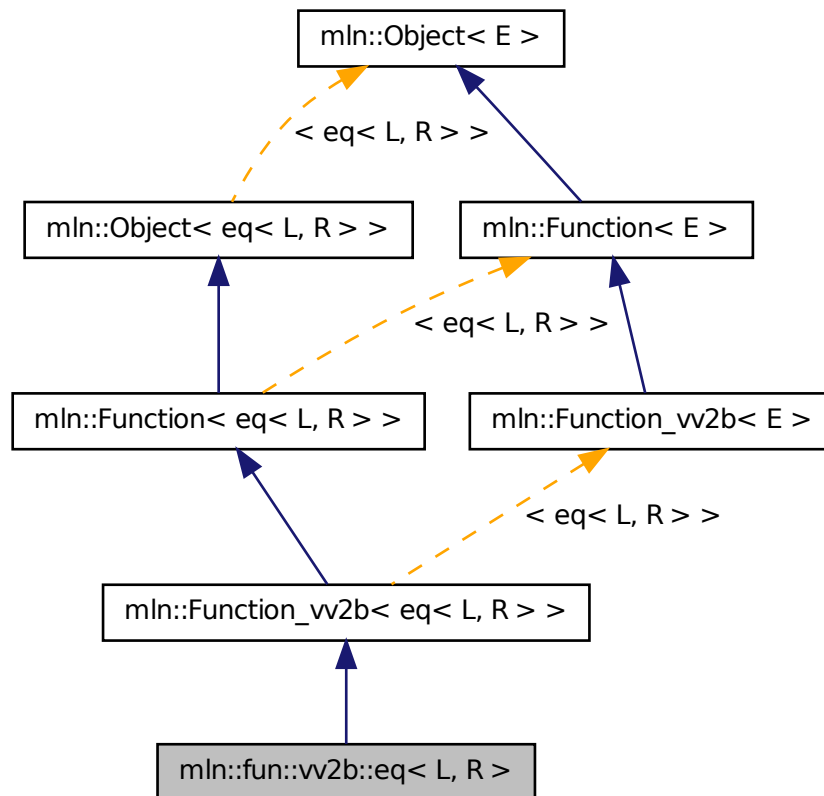
Definition at line 82 of file `v2w_w2v/norm.hh`.

10.158 mln::fun::vv2b::eq< L, R > Struct Template Reference

Functor computing equal between two values.

```
#include <eq.hh>
```

Inheritance diagram for mln::fun::vv2b::eq< L, R >:



10.158.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >
```

Functor computing equal between two values.

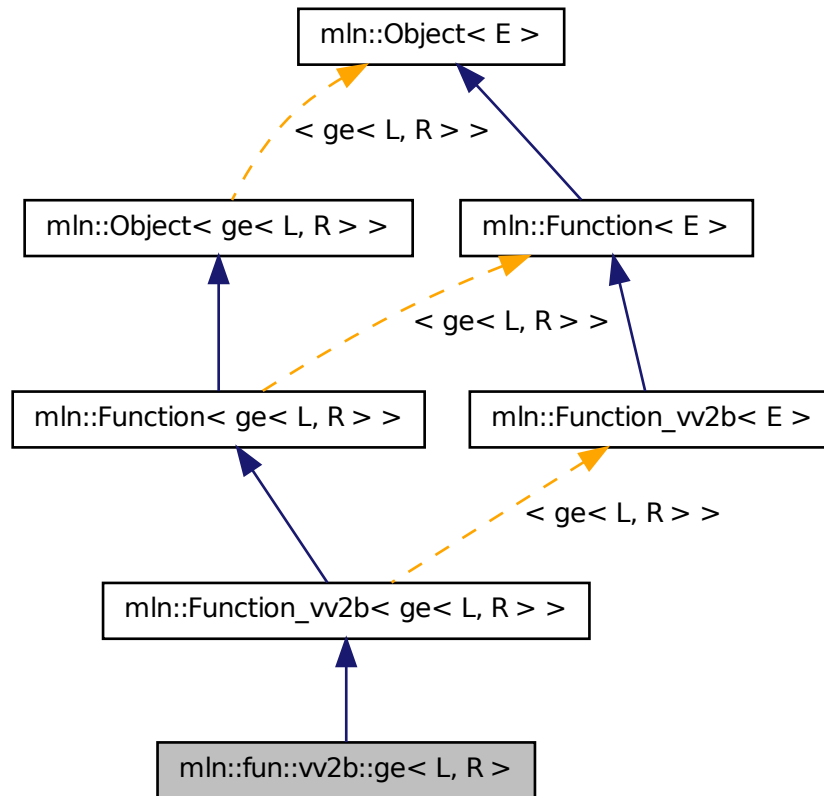
Definition at line 48 of file fun/vv2b/eq.hh.

10.159 mln::fun::vv2b::ge< L, R > Struct Template Reference

Functor computing "greater or equal than" between two values.

```
#include <ge.hh>
```

Inheritance diagram for mln::fun::vv2b::ge< L, R >:



10.159.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >
```

Functor computing "greater or equal than" between two values.

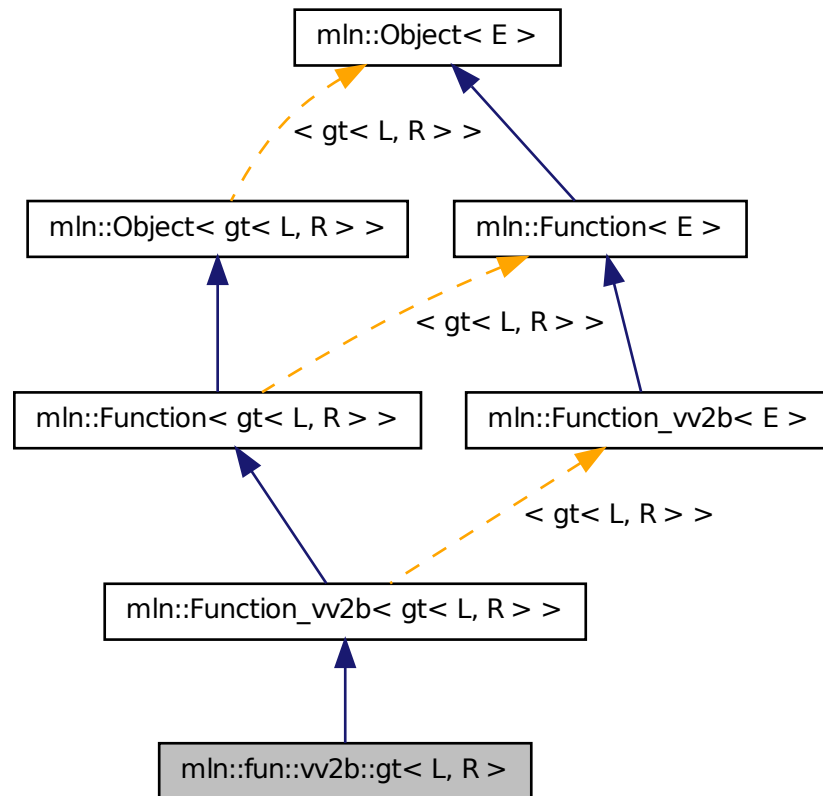
Definition at line 48 of file `ge.hh`.

10.160 mln::fun::vv2b::gt< L, R > Struct Template Reference

Functor computing "greater than" between two values.

```
#include <gt.hh>
```

Inheritance diagram for `mln::fun::vv2b::gt< L, R >`:



10.160.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >
```

Functor computing "greater than" between two values.

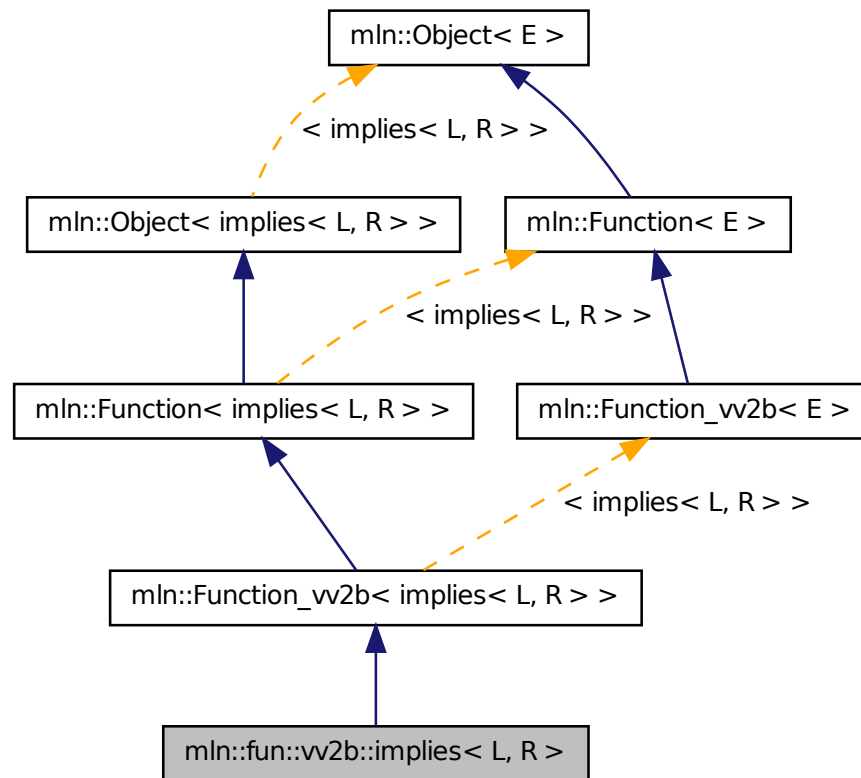
Definition at line 48 of file `gt.hh`.

10.161 `mln::fun::vv2b::implies< L, R >` Struct Template Reference

Functor computing logical-implies between two values.

```
#include <implies.hh>
```

Inheritance diagram for mln::fun::vv2b::implies< L, R >:



10.161.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >
```

Functor computing logical-implies between two values.

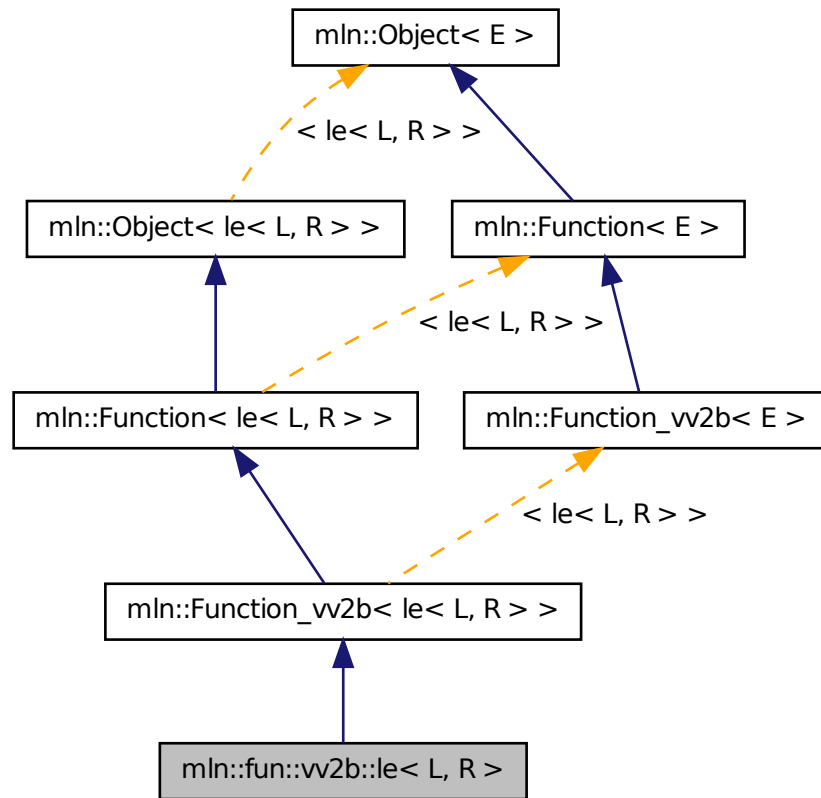
Definition at line 48 of file `implies.hh`.

10.162 mln::fun::vv2b::le< L, R > Struct Template Reference

Functor computing "lower or equal than" between two values.

```
#include <le.hh>
```

Inheritance diagram for `mln::fun::vv2b::le< L, R >`:



10.162.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >
```

Functor computing "lower or equal than" between two values.

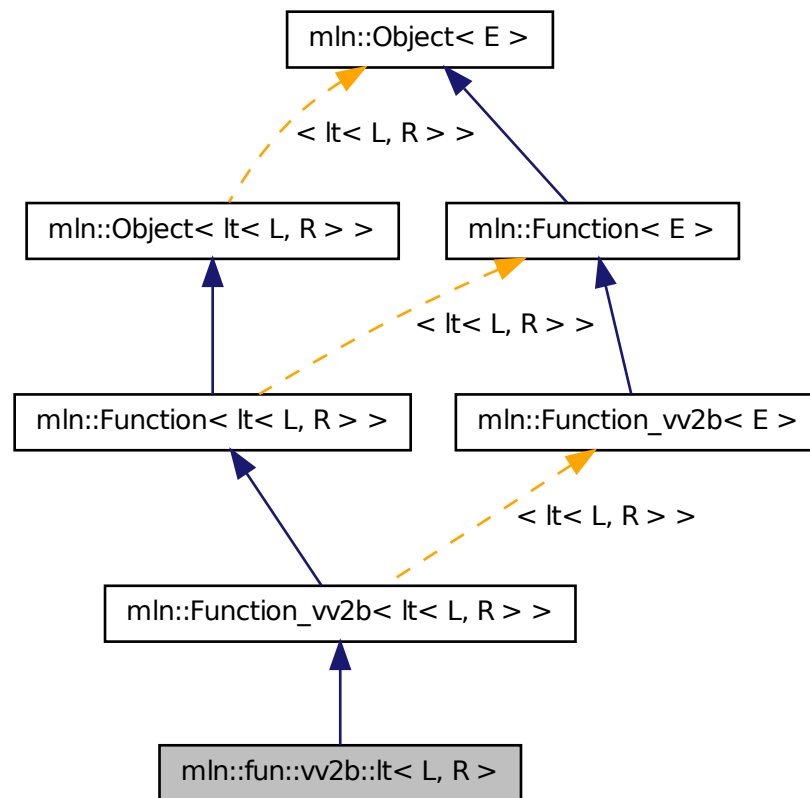
Definition at line 48 of file `le.hh`.

10.163 `mln::fun::vv2b::lt< L, R >` Struct Template Reference

Functor computing "lower than" between two values.

```
#include <lt.hh>
```


Inheritance diagram for mln::fun::vv2b::lt< L, R >:



10.163.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >
```

Functor computing "lower than" between two values.

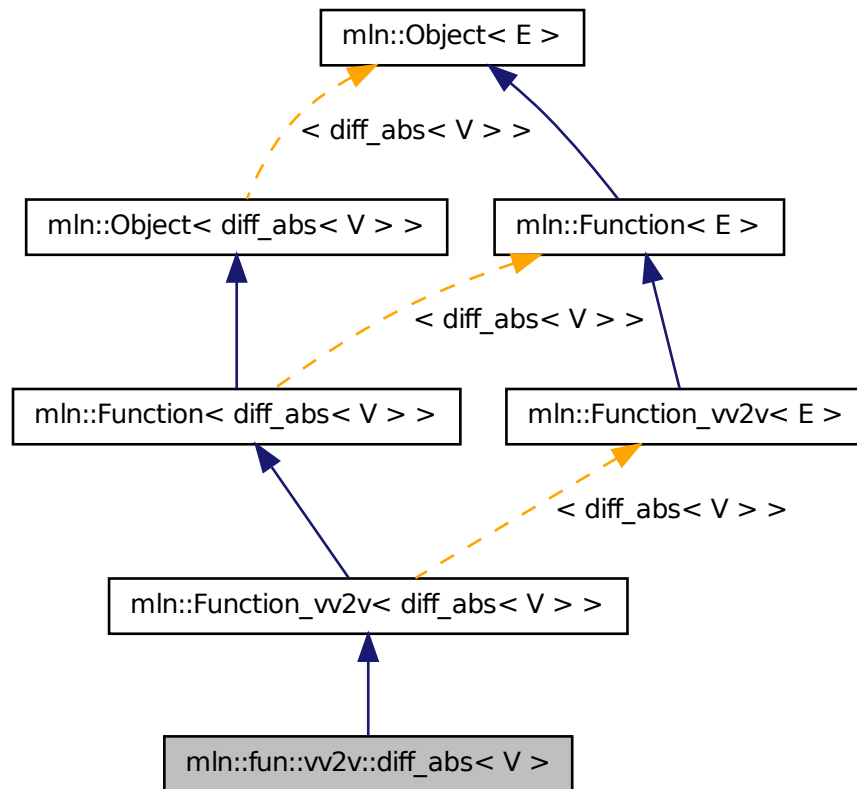
Definition at line 48 of file lt.hh.

10.164 mln::fun::vv2v::diff_abs< V > Struct Template Reference

A functor computing the `diff_absimum` of two values.

```
#include <diff_abs.hh>
```

Inheritance diagram for `mln::fun::vv2v::diff_abs< V >`:



10.164.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::diff_abs< V >
```

A functor computing the `diff_absimum` of two values.

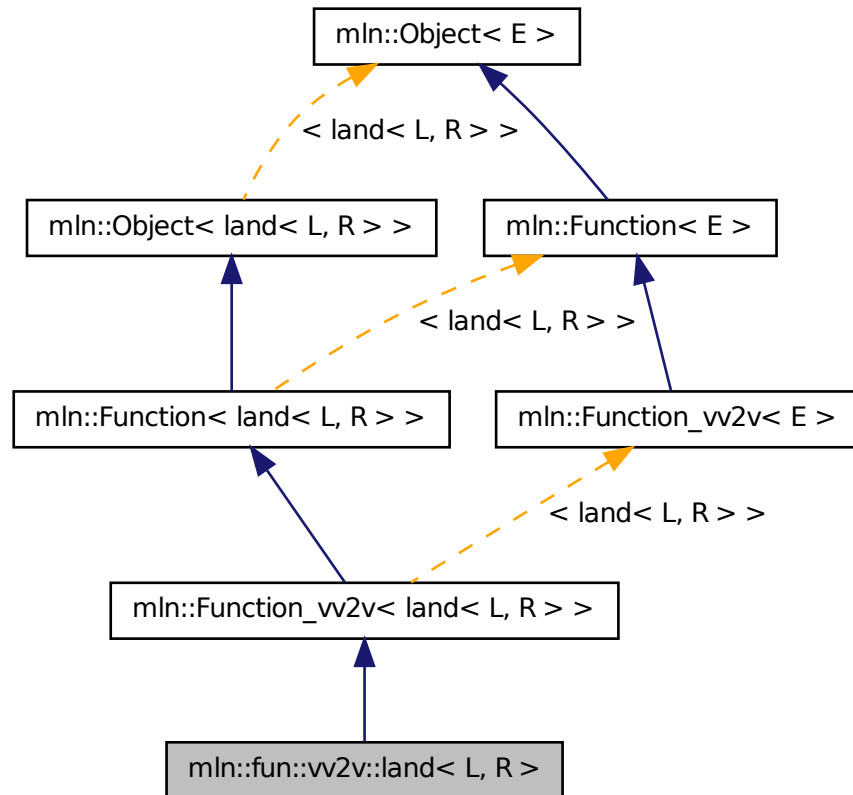
Definition at line 50 of file `fun/vv2v/diff_abs.hh`.

10.165 `mln::fun::vv2v::land< L, R >` Struct Template Reference

Functor computing logical-and between two values.

```
#include <land.hh>
```

Inheritance diagram for mln::fun::vv2v::land< L, R >:



10.165.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >
```

Functor computing logical-and between two values.

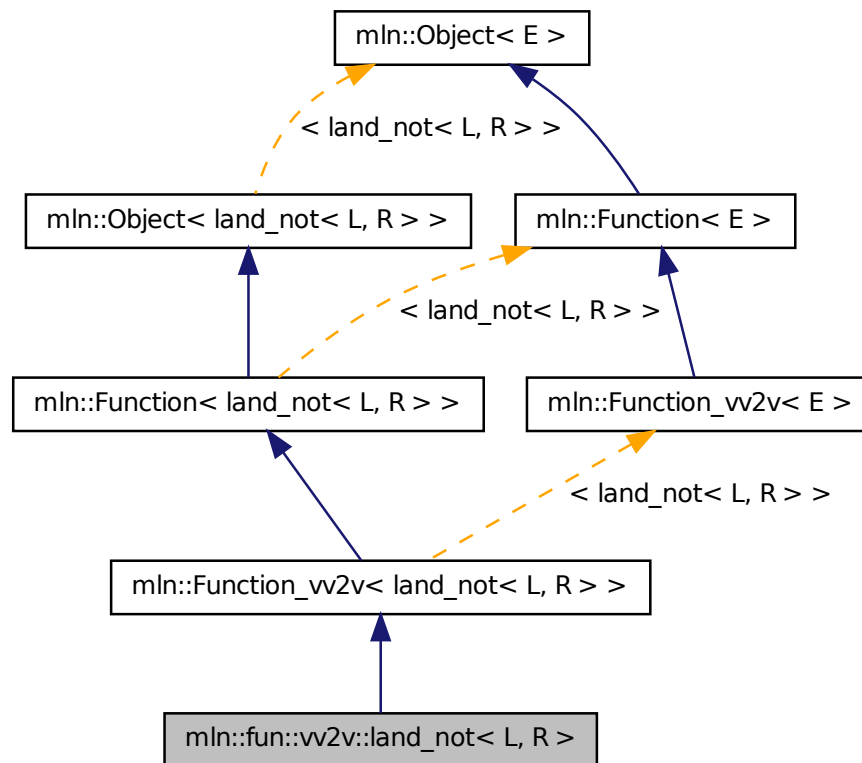
Definition at line 48 of file fun/vv2v/land.hh.

10.166 mln::fun::vv2v::land_not< L, R > Struct Template Reference

Functor computing logical and-not between two values.

```
#include <land_not.hh>
```

Inheritance diagram for `mln::fun::vv2v::land_not< L, R >`:



10.166.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >
```

Functor computing logical and-not between two values.

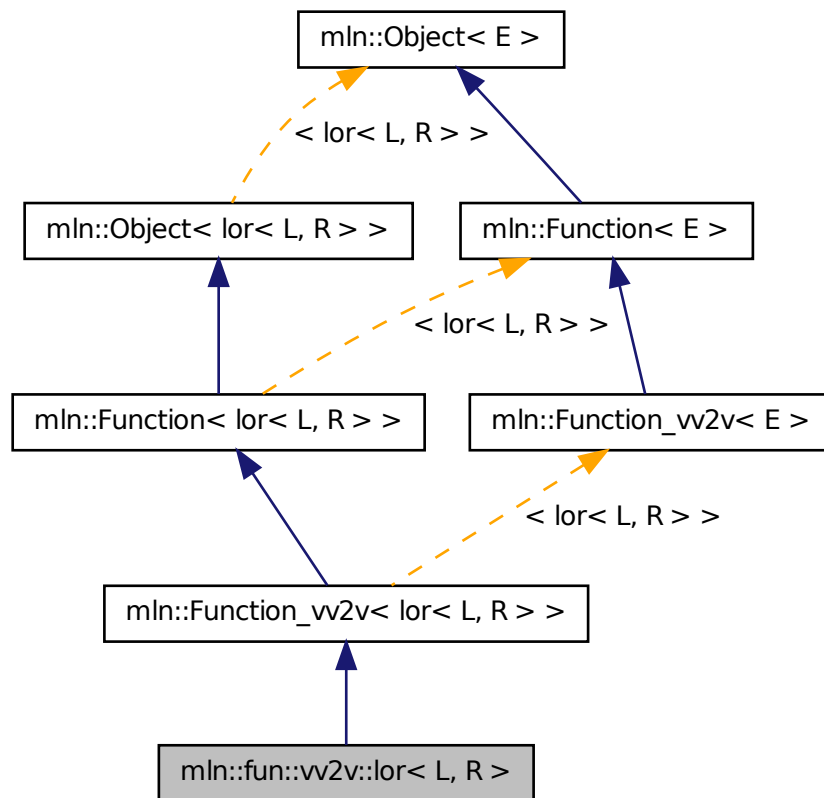
Definition at line 49 of file `land_not.hh`.

10.167 mln::fun::vv2v::lor< L, R > Struct Template Reference

Functor computing logical-or between two values.

```
#include <lor.hh>
```

Inheritance diagram for mln::fun::vv2v::lor< L, R >:



10.167.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >
```

Functor computing logical-or between two values.

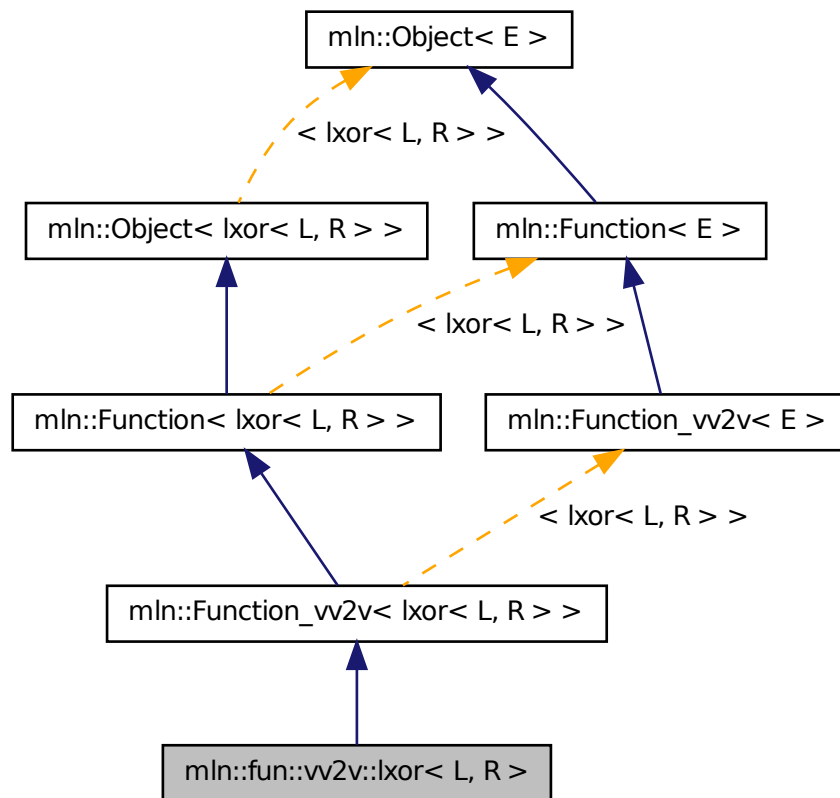
Definition at line 48 of file fun/vv2v/lor.hh.

10.168 mln::fun::vv2v::lxor< L, R > Struct Template Reference

Functor computing logical-xor between two values.

```
#include <lxor.hh>
```

Inheritance diagram for `mln::fun::vv2v::lxor< L, R >`:



10.168.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >
```

Functor computing logical-xor between two values.

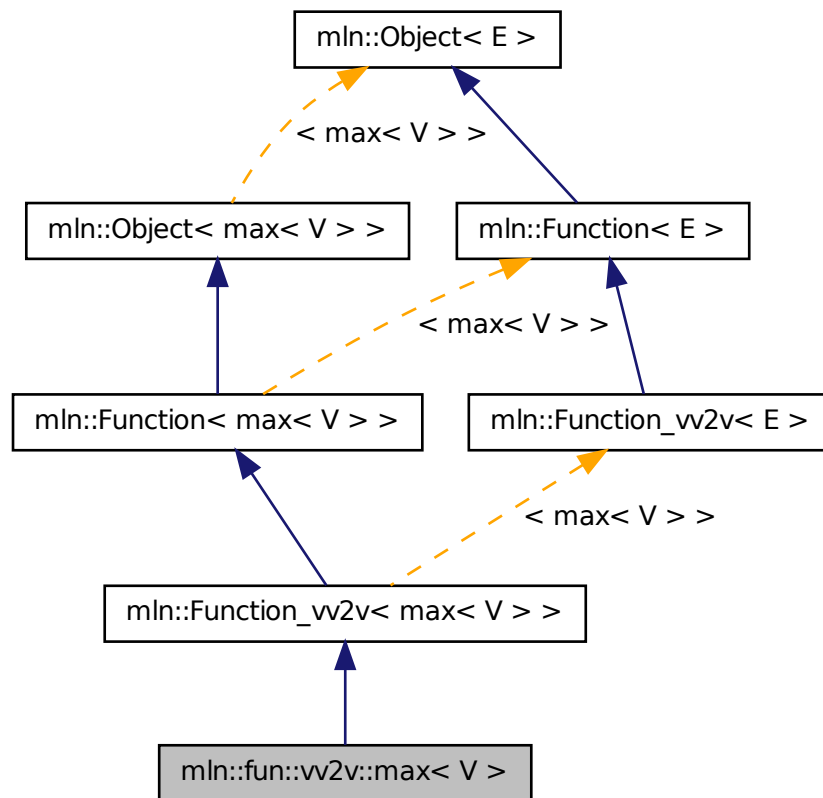
Definition at line 48 of file `lxor.hh`.

10.169 mln::fun::vv2v::max< V > Struct Template Reference

A functor computing the maximum of two values.

```
#include <max.hh>
```

Inheritance diagram for mln::fun::vv2v::max< V >:



10.169.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::max< V >
```

A functor computing the maximum of two values.

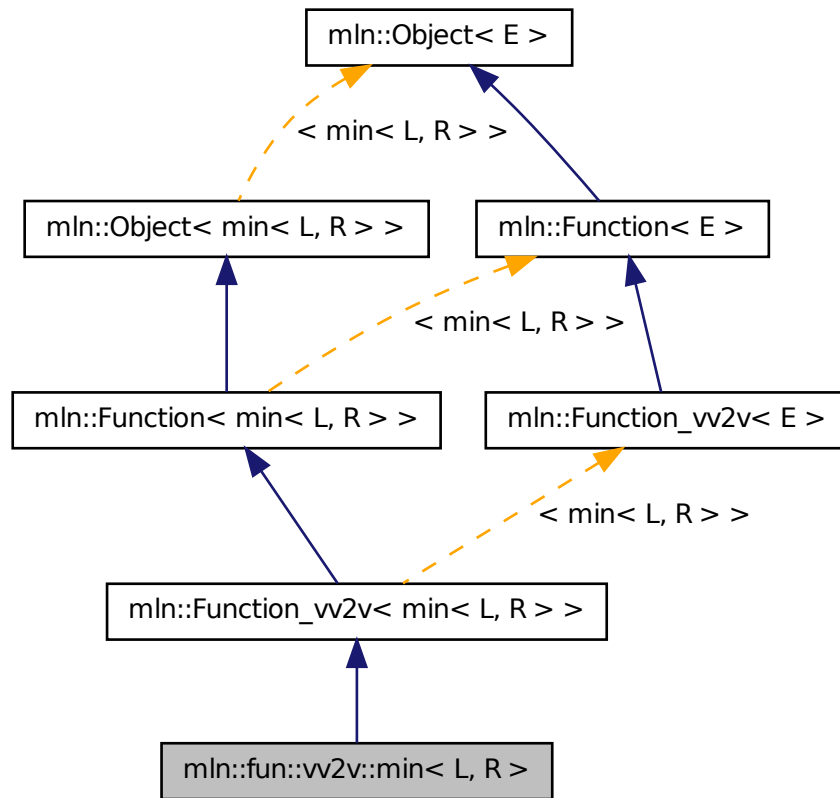
Definition at line 49 of file `fun/vv2v/max.hh`.

10.170 mln::fun::vv2v::min< L, R > Struct Template Reference

A functor computing the minimum of two values.

```
#include <min.hh>
```

Inheritance diagram for `mln::fun::vv2v::min< L, R >`:



10.170.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >
```

A functor computing the minimum of two values.

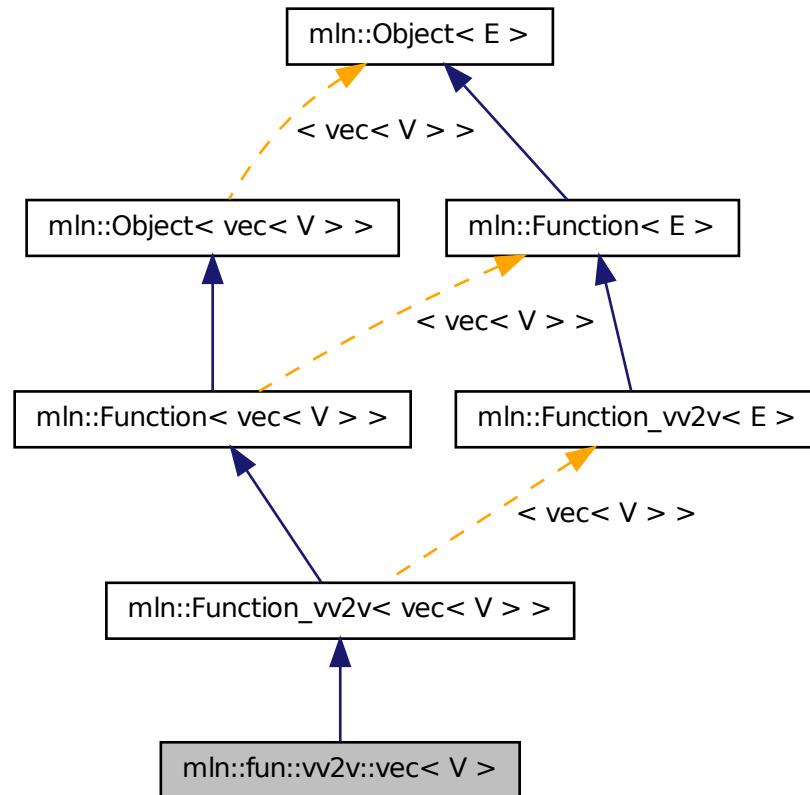
Definition at line 50 of file `fun/vv2v/min.hh`.

10.171 mln::fun::vv2v::vec< V > Struct Template Reference

A functor computing the vecimum of two values.

```
#include <vec.hh>
```


Inheritance diagram for mln::fun::vv2v::vec< V >:



10.171.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::vec< V >
```

A functor computing the vecimum of two values.

Definition at line 50 of file fun/vv2v/vec.hh.

10.172 mln::fun::x2p::closest_point< P > Struct Template Reference

FIXME: doxygen + concept checking.

```
#include <closest_point.hh>
```

10.172.1 Detailed Description

`template<typename P> struct mln::fun::x2p::closest_point< P >`

FIXME: doxygen + concept checking.

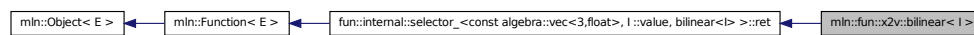
Definition at line 45 of file `closest_point.hh`.

10.173 mln::fun::x2v::bilinear< I > Struct Template Reference

Represent a bilinear interolation of values from an underlying image.

`#include <bilinear.hh>`

Inheritance diagram for `mln::fun::x2v::bilinear< I >`:



Public Member Functions

- `template<typename T >`
`I::value operator()` (`const algebra::vec< 3, T > &v`) `const`
Bilinear filtering on 3d images. Work on slices.
- `template<typename T >`
`I::value operator()` (`const algebra::vec< 2, T > &v`) `const`
Bilinear filtering on 2d images.

10.173.1 Detailed Description

`template<typename I> struct mln::fun::x2v::bilinear< I >`

Represent a bilinear interolation of values from an underlying image.

Definition at line 52 of file `bilinear.hh`.

10.173.2 Member Function Documentation

10.173.2.1 `template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator()` (`const algebra::vec< 2, T > & v`) `const`

Bilinear filtering on 2d images.

Definition at line 86 of file `bilinear.hh`.

10.173.2.2 `template<typename I> template<typename T> I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 3, T > & v) const`

Bilinear filtering on 3d images. Work on slices.

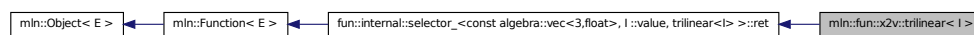
Definition at line 132 of file bilinear.hh.

10.174 `mln::fun::x2v::trilinear< I > Struct Template Reference`

Represent a trilinear interolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for `mln::fun::x2v::trilinear< I >`:



10.174.1 Detailed Description

`template<typename I> struct mln::fun::x2v::trilinear< I >`

Represent a trilinear interolation of values from an underlying image.

Definition at line 53 of file trilinear.hh.

10.175 `mln::fun::x2x::composed< T2, T1 > Struct Template Reference`

Represent a composition of two transformations.

```
#include <composed.hh>
```

Public Member Functions

- `composed()`
Constructor without argument.
- `composed(const T2 &f, const T1 &g)`
Constructor with the two transformation to be composed.

10.175.1 Detailed Description

`template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >`

Represent a composition of two transformations.

Definition at line 144 of file composed.hh.

10.175.2 Constructor & Destructor Documentation

10.175.2.1 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed () [inline]`

Constructor without argument.

Definition at line 153 of file composed.hh.

10.175.2.2 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed (const T2 & f, const T1 & g) [inline]`

Constructor with the two transformation to be composed.

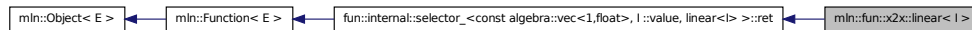
Definition at line 156 of file composed.hh.

10.176 mln::fun::x2x::linear< I > Struct Template Reference

Represent a linear interolation of values from an underlying image.

```
#include <linear.hh>
```

Inheritance diagram for mln::fun::x2x::linear< I >:



Public Member Functions

- `linear` (const I & [ima](#))

Constructor with the underlying image.

- `template<typename C >`
`I::value` `operator()` (const algebra::vec< 1, C > &v) const

Return the interpolated value in the underlying image at the given 'point' v.

Public Attributes

- const I & [ima](#)

Underlying image.

10.176.1 Detailed Description

template<typename I> struct mln::fun::x2x::linear< I >

Represent a linear interolation of values from an underlying image.

Definition at line 53 of file x2v/linear.hh.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 template<typename I> mln::fun::x2x::linear< I >::linear (const I & *ima*)

Constructor with the underlying image.

Definition at line 77 of file x2v/linear.hh.

10.176.3 Member Function Documentation

10.176.3.1 template<typename I> template<typename C> I::value mln::fun::x2x::linear< I >::operator() (const algebra::vec< 1, C > & *v*) const

Return the interpolated value in the underlying image at the given 'point' *v*.

Definition at line 85 of file x2v/linear.hh.

10.176.4 Member Data Documentation

10.176.4.1 template<typename I> const I& mln::fun::x2x::linear< I >::ima

Underlying image.

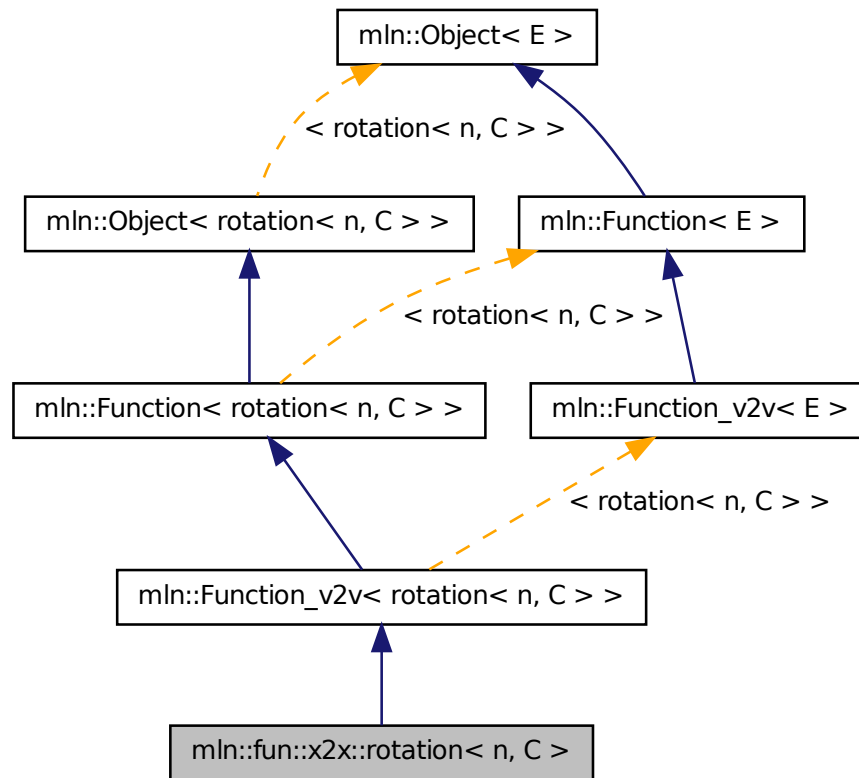
Definition at line 70 of file x2v/linear.hh.

10.177 mln::fun::x2x::rotation< n, C > Struct Template Reference

Represent a rotation function.

#include <rotation.hh>

Inheritance diagram for `mln::fun::x2x::rotation< n, C >`:



Public Types

- typedef `C` [data_t](#)
Type of the underlying data stored in vectors and matrices.
- typedef [rotation< n, C >](#) [invert](#)
Type of the inverse function.

Public Member Functions

- [invert](#) [inv](#) () const
Return the inverse function.
- `algebra::vec< n, C >` [operator\(\)](#) (const `algebra::vec< n, C >` &`v`) const
Perform the rotation of the given vector.

- [rotation](#) ()
Constructor without argument.
- [rotation](#) (const algebra::quat &q)
Constructor with quaternion.
- [rotation](#) (const algebra::h_mat< n, C > &m)
Constructor with h_mat.
- [rotation](#) (C alpha, const algebra::vec< n, C > &axis)
Constructor with radian alpha and a facultative direction (rotation axis).
- void [set_alpha](#) (C alpha)
Set a new grade alpha.
- void [set_axis](#) (const algebra::vec< n, C > &axis)
Set a new rotation axis.

10.177.1 Detailed Description

template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >

Represent a rotation function.

Definition at line 147 of file rotation.hh.

10.177.2 Member Typedef Documentation

10.177.2.1 template<unsigned n, typename C > typedef C mln::fun::x2x::rotation< n, C >::data_t

Type of the underlying data stored in vectors and matrices.

Definition at line 152 of file rotation.hh.

10.177.2.2 template<unsigned n, typename C > typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert

Type of the inverse function.

Definition at line 155 of file rotation.hh.

10.177.3 Constructor & Destructor Documentation

**10.177.3.1 template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ()
[inline]**

Constructor without argument.

Definition at line 192 of file rotation.hh.

10.177.3.2 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (C
alpha, const algebra::vec< n, C > & axis) [inline]`

Constructor with radian alpha and a facultative direction (rotation axis).

Definition at line 198 of file rotation.hh.

10.177.3.3 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (
const algebra::quat & q) [inline]`

Constructor with quaternion.

Definition at line 208 of file rotation.hh.

References mln::make::h_mat().

10.177.3.4 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (
const algebra::h_mat< n, C > & m) [inline]`

Constructor with h_mat.

Definition at line 238 of file rotation.hh.

10.177.4 Member Function Documentation

10.177.4.1 `template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C
>::inv () const [inline]`

Return the inverse function.

Definition at line 265 of file rotation.hh.

10.177.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::rotation< n, C
>::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the rotation of the given vector.

Definition at line 247 of file rotation.hh.

10.177.4.3 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha (
C alpha) [inline]`

Set a new grade alpha.

Definition at line 274 of file rotation.hh.

10.177.4.4 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_axis (
const algebra::vec< n, C > & axis) [inline]`

Set a new rotation axis.

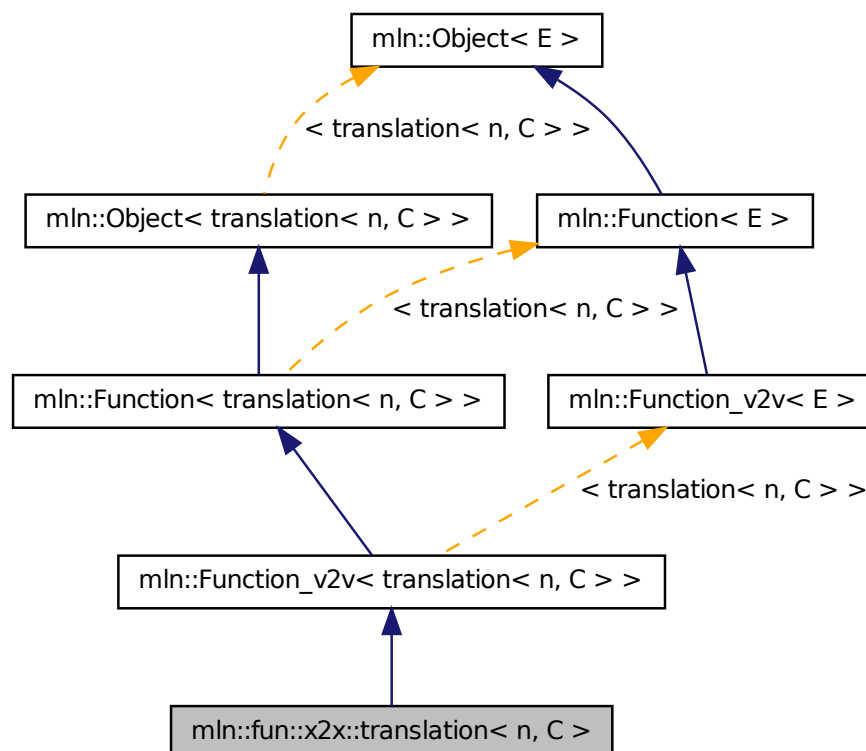
Definition at line 283 of file rotation.hh.

10.178 mln::fun::x2x::translation< n, C > Struct Template Reference

Translation function-object.

```
#include <translation.hh>
```

Inheritance diagram for mln::fun::x2x::translation< n, C >:



Public Types

- typedef C `data_t`
Type of the underlying data stored in vectors and matrices.
- typedef `translation< n, C >` `invert`
Type of the inverse function.

Public Member Functions

- `invert inv () const`

Return the inverse function.

- `algebra::vec< n, C > operator()` (`const algebra::vec< n, C > &v`) `const`

Perform the translation of the given vector.

- `void set_t` (`const algebra::vec< n, C > &t`)

Set a net translation vector.

- `const algebra::vec< n, C > & t ()` `const`

Return the translation vector.

- `translation` (`const algebra::vec< n, C > &t`)

Constructor with the translation vector.

- `translation ()`

Constructor without argument.

10.178.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >`

Translation function-object.

Definition at line 52 of file `x2x/translation.hh`.

10.178.2 Member Typedef Documentation

10.178.2.1 `template<unsigned n, typename C > typedef C mln::fun::x2x::translation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

Definition at line 59 of file `x2x/translation.hh`.

10.178.2.2 `template<unsigned n, typename C > typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert`

Type of the inverse function.

Definition at line 62 of file `x2x/translation.hh`.

10.178.3 Constructor & Destructor Documentation

10.178.3.1 `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation () [inline]`

Constructor without argument.

Definition at line 93 of file `x2x/translation.hh`.

10.178.3.2 `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation (const algebra::vec< n, C > & t) [inline]`

Constructor with the translation vector.

Definition at line 99 of file x2x/translation.hh.

10.178.4 Member Function Documentation

10.178.4.1 `template<unsigned n, typename C > translation< n, C > mln::fun::x2x::translation< n, C >::inv () const [inline]`

Return the inverse function.

Definition at line 124 of file x2x/translation.hh.

10.178.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::translation< n, C >::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the translation of the given vector.

Definition at line 108 of file x2x/translation.hh.

10.178.4.3 `template<unsigned n, typename C > void mln::fun::x2x::translation< n, C >::set_t (const algebra::vec< n, C > & t) [inline]`

Set a net translation vector.

Definition at line 134 of file x2x/translation.hh.

10.178.4.4 `template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t () const [inline]`

Return the translation vector.

Definition at line 143 of file x2x/translation.hh.

10.179 mln::fun_image< F, I > Struct Template Reference

[Image](#) read through a function.

```
#include <fun_image.hh>
```

Inherits `image_value_morpher< I, F::result, fun_image< F, I > >`.

Public Types

- typedef F::result [lvalue](#)
Return type of read-write access.
- typedef F::result [rvalue](#)

Return type of read-only access.

- typedef [fun_image](#)< tag::value_< typename F::result >, tag::image_< I > > [skeleton](#)
Skeleton.
- typedef F::result [value](#)
Value associated type.

Public Member Functions

- [fun_image](#) ()
Constructor.
- [fun_image](#) (const [Function_v2v](#)< F > &f, const [Image](#)< I > &ima)
Constructor.
- [fun_image](#) (const [Image](#)< I > &ima)
Constructor.
- F::result [operator](#)() (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- F::result [operator](#)() (const typename I::psite &p)
Mutable access is for reading only.

10.179.1 Detailed Description

template<typename F, typename I> struct mln::fun_image< F, I >

[Image](#) read through a function.

Definition at line 101 of file fun_image.hh.

10.179.2 Member Typedef Documentation

10.179.2.1 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue

Return type of read-write access.

Definition at line 111 of file fun_image.hh.

10.179.2.2 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue

Return type of read-only access.

Definition at line 108 of file fun_image.hh.

10.179.2.3 `template<typename F, typename I> typedef fun_image< tag::value_<typename F
::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton`

Skeleton.

Definition at line 115 of file fun_image.hh.

10.179.2.4 `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value`

[Value](#) associated type.

Definition at line 105 of file fun_image.hh.

10.179.3 Constructor & Destructor Documentation

10.179.3.1 `template<typename F , typename I > mln::fun_image< F, I >::fun_image ()
[inline]`

Constructor.

Definition at line 177 of file fun_image.hh.

10.179.3.2 `template<typename F , typename I > mln::fun_image< F, I >::fun_image (const
Function_v2v< F > & f, const Image< I > & ima) [inline]`

Constructor.

Definition at line 184 of file fun_image.hh.

10.179.3.3 `template<typename F , typename I > mln::fun_image< F, I >::fun_image (const
Image< I > & ima) [inline]`

Constructor.

Definition at line 191 of file fun_image.hh.

10.179.4 Member Function Documentation

10.179.4.1 `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() (const
typename I::psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 209 of file fun_image.hh.

10.179.4.2 `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() (const
typename I::psite & p) [inline]`

Mutable access is for reading only.

Definition at line 218 of file fun_image.hh.

10.180 mln::Function< E > Struct Template Reference

Base class for implementation of function-objects.

```
#include <function.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Function_n2v< E >](#), [mln::Function_v2v< E >](#), [mln::Function_vv2b< E >](#), and [mln::Function_vv2v< E >](#).

Protected Member Functions

- [Function](#) ()

An operator() has to be provided.

10.180.1 Detailed Description

```
template<typename E> struct mln::Function< E >
```

Base class for implementation of function-objects. The parameter *E* is the exact type.

Definition at line 64 of file function.hh.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 `template<typename E > mln::Function< E >::Function () [inline, protected]`

An operator() has to be provided.

Its signature depends on the particular function-object one considers.

Definition at line 219 of file function.hh.

10.181 mln::Function< void > Struct Template Reference

[Function](#) category flag type.

```
#include <function.hh>
```

10.181.1 Detailed Description

```
template<> struct mln::Function< void >
```

[Function](#) category flag type.

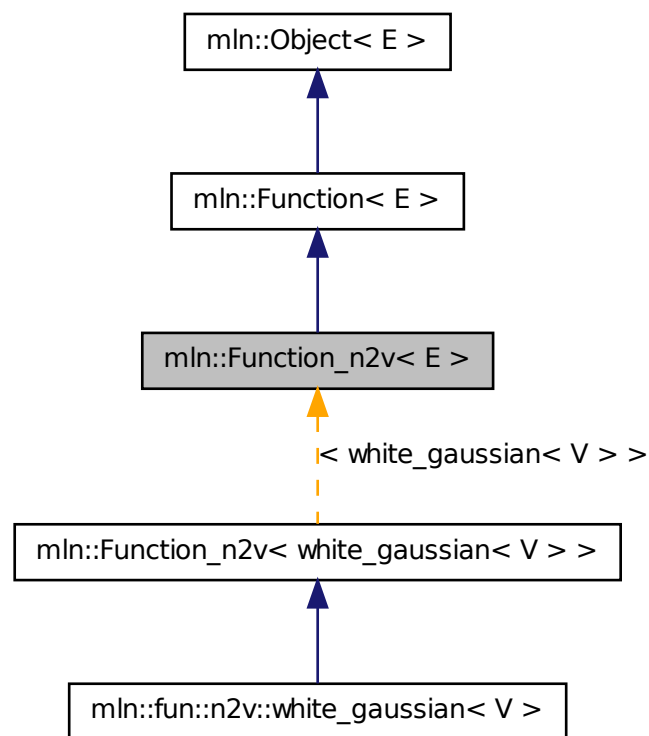
Definition at line 51 of file function.hh.

10.182 mln::Function_n2v< E > Struct Template Reference

Base class for implementation of function-objects from Nil to value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function_n2v< E >:



10.182.1 Detailed Description

```
template<typename E> struct mln::Function_n2v< E >
```

Base class for implementation of function-objects from Nil to value. The parameter *E* is the exact type.

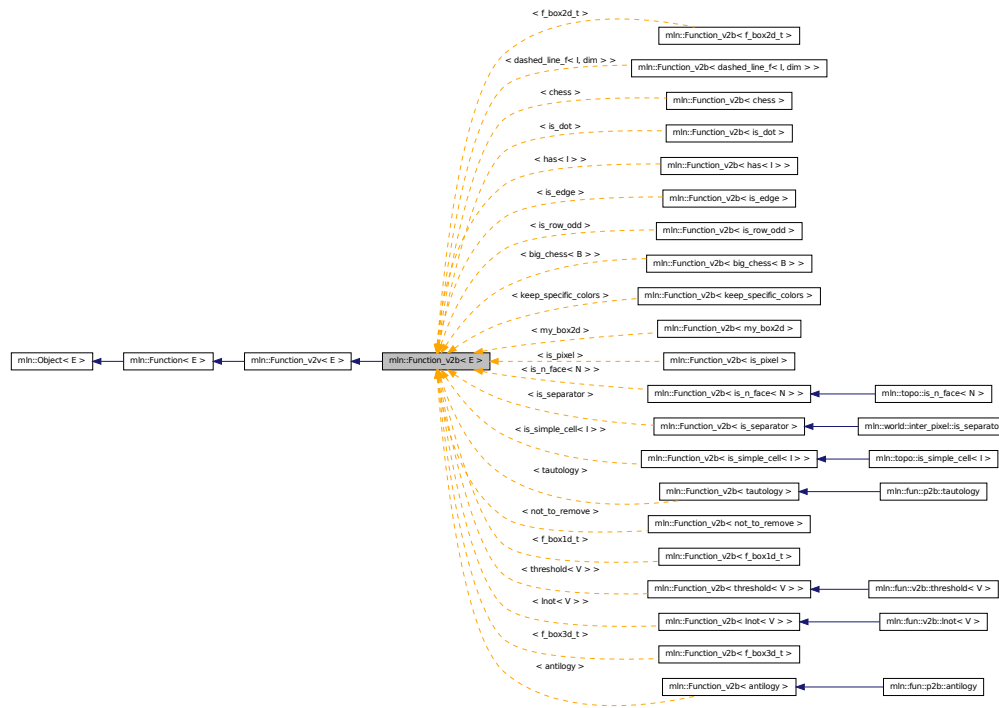
Definition at line 99 of file `function.hh`.

10.183 mln::Function_v2b< E > Struct Template Reference

Base class for implementation of function-objects from a value to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for `mln::Function_v2b< E >`:



10.183.1 Detailed Description

`template<typename E> struct mln::Function_v2b< E >`

Base class for implementation of function-objects from a value to a Boolean. The parameter *E* is the exact type.

Definition at line 150 of file `function.hh`.

10.184 mln::Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

`#include <function.hh>`

Inherits [mln::Function< E >](#).

Inherited by `mln::fun::C< R(*) (A) >`, `mln::fun::v2v::dec< T >`, `mln::fun::v2v::id< T >`, `mln::fun::v2v::inc< T >`, `mln::fun::x2v::bilinear< I >`, `mln::fun::x2v::trilinear< I >`, `mln::fun::x2x::internal::helper_composed_< T2, T1, E, false >`, `mln::fun::x2x::internal::helper_composed_< T2, T1, E, true >`, `mln::fun::x2x::linear< I >`, `mln::fun::x2x::neighbor< I >`, and `mln::Function_v2b< E >` [virtual].

10.184.1 Detailed Description

template<typename E> struct mln::Function_v2v< E >

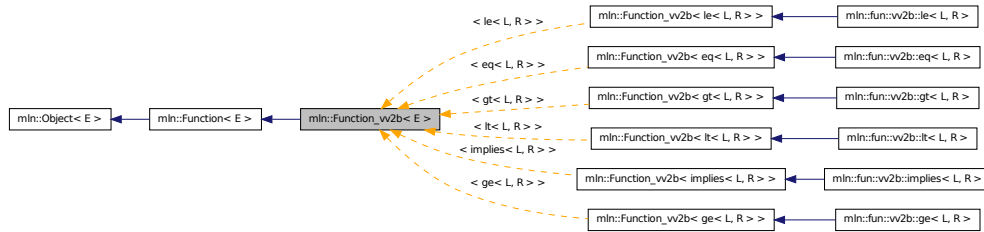
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type. Definition at line 124 of file function.hh.

10.185 mln::Function_vv2b< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for mln::Function_vv2b< E >:



10.185.1 Detailed Description

template<typename E> struct mln::Function_vv2b< E >

Base class for implementation of function-objects from a couple of values to a Boolean. The parameter *E* is the exact type.

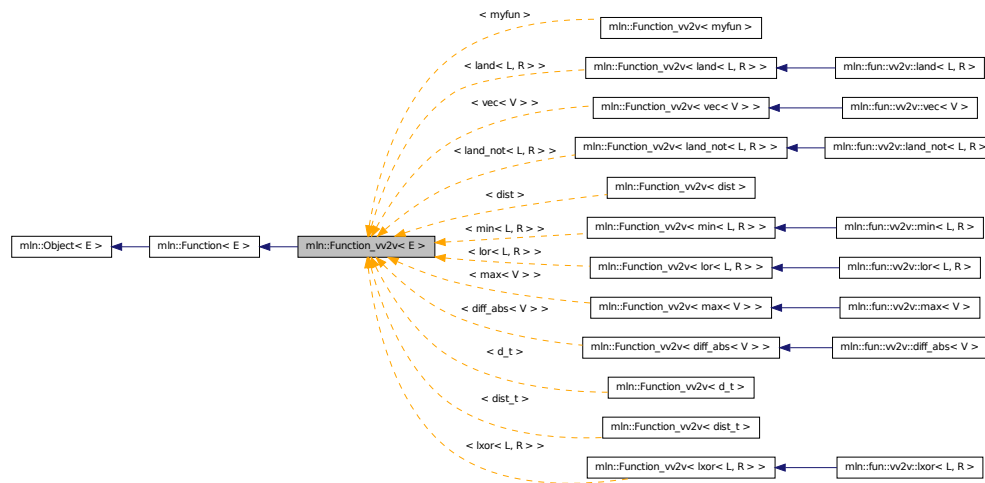
Definition at line 202 of file function.hh.

10.186 mln::Function_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a value.

```
#include <function.hh>
```

Inheritance diagram for `mln::Function_vv2v< E >`:



10.186.1 Detailed Description

template<typename E> struct mln::Function_vv2v< E >

Base class for implementation of function-objects from a couple of values to a value. The parameter *E* is the exact type.

Definition at line 177 of file `function.hh`.

10.187 mln::fwd_pixter1d< I > Class Template Reference

Forward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits `forward_pixel_iterator_base_< I, fwd_pixter1d< I > >`.

Public Types

- typedef `I` [image](#)
Image type.

Public Member Functions

- `fwd_pixter1d` (`I &image`)
Constructor.
- void `next` ()

Go to the next element.

10.187.1 Detailed Description

template<typename I> class mln::fwd_pixter1d< I >

Forward pixel iterator on a 1-D image with border.

Definition at line 45 of file pixter1d.hh.

10.187.2 Member Typedef Documentation

10.187.2.1 template<typename I > typedef I mln::fwd_pixter1d< I >::image

[Image](#) type.

Definition at line 52 of file pixter1d.hh.

10.187.3 Constructor & Destructor Documentation

10.187.3.1 template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d (I & *image*)
[inline]

Constructor.

Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 96 of file pixter1d.hh.

10.187.4 Member Function Documentation

10.187.4.1 void mln::Iterator< fwd_pixter1d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.188 mln::fwd_pixter2d< I > Class Template Reference

Forward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `forward_pixel_iterator_base_< I, fwd_pixter2d< I > >`.

Public Types

- typedef `I` `image`
Image type.

Public Member Functions

- `fwd_pixter2d` (`I &image`)
Constructor.
- void `next` ()
Go to the next element.

10.188.1 Detailed Description

```
template<typename I> class mln::fwd_pixter2d< I >
```

Forward pixel iterator on a 2-D image with border.

Definition at line 47 of file `pixter2d.hh`.

10.188.2 Member Typedef Documentation

10.188.2.1 `template<typename I > typedef I mln::fwd_pixter2d< I >::image`

`Image` type.

Definition at line 54 of file `pixter2d.hh`.

10.188.3 Constructor & Destructor Documentation

10.188.3.1 `template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 130 of file `pixter2d.hh`.

10.188.4 Member Function Documentation

10.188.4.1 void mln::Iterator< fwd_pixter2d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.189 mln::fwd_pixter3d< I > Class Template Reference

Forward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits forward_pixel_iterator_base_< I, fwd_pixter3d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [fwd_pixter3d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.189.1 Detailed Description

```
template<typename I> class mln::fwd_pixter3d< I >
```

Forward pixel iterator on a 3-D image with border.

Definition at line 48 of file pixter3d.hh.

10.189.2 Member Typedef Documentation

10.189.2.1 template<typename I > typedef I mln::fwd_pixter3d< I >::image

[Image](#) type.

Definition at line 55 of file pixter3d.hh.

10.189.3 Constructor & Destructor Documentation

10.189.3.1 `template<typename I> mln::fwd_pixter3d<I>::fwd_pixter3d (I & image)`
[inline]

Constructor.

Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 154 of file pixter3d.hh.

10.189.4 Member Function Documentation

10.189.4.1 `void mln::Iterator< fwd_pixter3d<I> >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

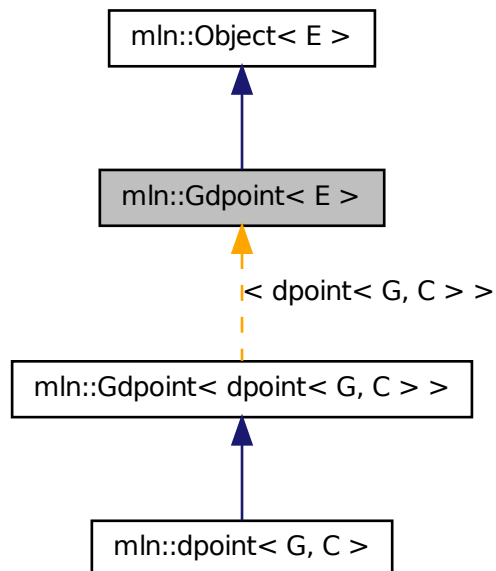
The iterator is valid.

10.190 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for mln::Gdpoint< E >:



10.190.1 Detailed Description

```
template<typename E> struct mln::Gdpoint< E >
```

FIXME: Doc!

Definition at line 95 of file `gdpoint.hh`.

10.191 mln::Gdpoint< void > Struct Template Reference

Delta point site category flag type.

```
#include <gdpoint.hh>
```

10.191.1 Detailed Description

```
template<> struct mln::Gdpoint< void >
```

Delta point site category flag type.

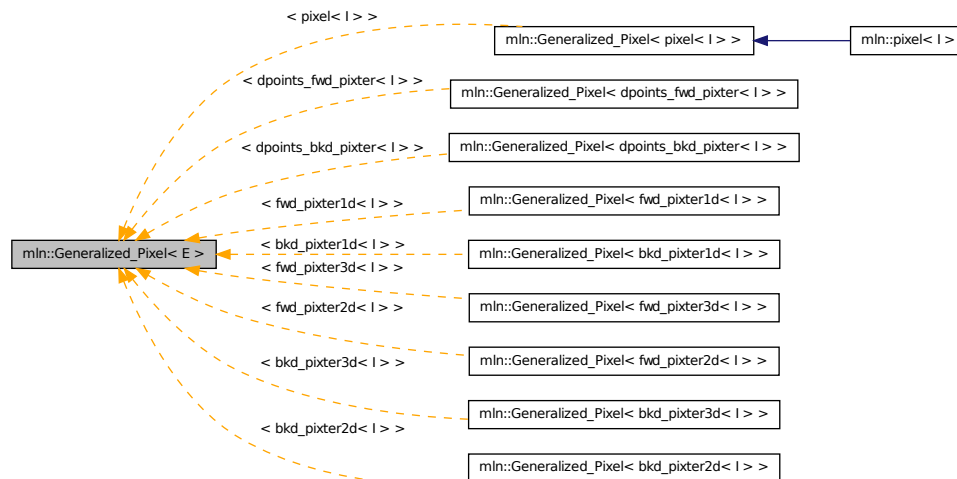
Definition at line 87 of file `gdpoint.hh`.

10.192 mln::Generalized_Pixel< E > Struct Template Reference

Base class for implementation classes that are pixels or that have the behavior of pixels.

```
#include <generalized_pixel.hh>
```

Inheritance diagram for mln::Generalized_Pixel< E >:



10.192.1 Detailed Description

```
template<typename E> struct mln::Generalized_Pixel< E >
```

Base class for implementation classes that are pixels or that have the behavior of pixels.

Warning

This class does *not* derive from [mln::Object](#); it is for use as a parallel hierarchy.

See also

[mln::doc::Generalized_Pixel](#) for a complete documentation of this class contents.

Definition at line 53 of file `generalized_pixel.hh`.

10.193 mln::geom::complex_geometry< D, P > Class Template Reference

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

```
#include <complex_geometry.hh>
```


Public Member Functions

- unsigned [add_location](#) (const P &p)
Populate the set of locations.
- [complex_geometry](#) ()
Build a complex geometry object.
- site [operator\(\)](#) (const [mln::topo::face](#)< D > &f) const
Retrieve the site associated to f.

10.193.1 Detailed Description

template<unsigned D, typename P> class mln::geom::complex_geometry< D, P >

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored. Faces of higher dimensions are computed.

Template Parameters

- D** The dimension of the complex.
P The type of the location of a 0-face.

Locations of 0-face are usually points (hence the P above), but can possibly be any (default-constructible) values.

The functor returns a std::vector of locations: 0-faces are singletons, 1-faces are (usually) pairs, faces of higher dimensions are arrays of locations.

Note that for consistency reasons w.r.t. the return type of operator(), returned sites are always *arrays* of locations attached to 0-faces; hence the returned singletons (of locations) for 0-faces.

Definition at line 88 of file geom/complex_geometry.hh.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 template<unsigned D, typename P > mln::geom::complex_geometry< D, P >::complex_geometry () [inline]

Build a complex geometry object.

Definition at line 132 of file geom/complex_geometry.hh.

10.193.3 Member Function Documentation

10.193.3.1 template<unsigned D, typename P > unsigned mln::geom::complex_geometry< D, P >::add_location (const P & p) [inline]

Populate the set of locations.

Append a new location p. Return the index of the newly created location (which should semantically match the id of the corresponding 0-face in the complex).

Definition at line 140 of file geom/complex_geometry.hh.

10.193.3.2 `template<unsigned D, typename P > util::multi_site< P > mln::geom::complex_geometry< D, P >::operator() (const mln::topo::face< D > & f) const`
[inline]

Retrieve the site associated to *f*.

Definition at line 151 of file `geom/complex_geometry.hh`.

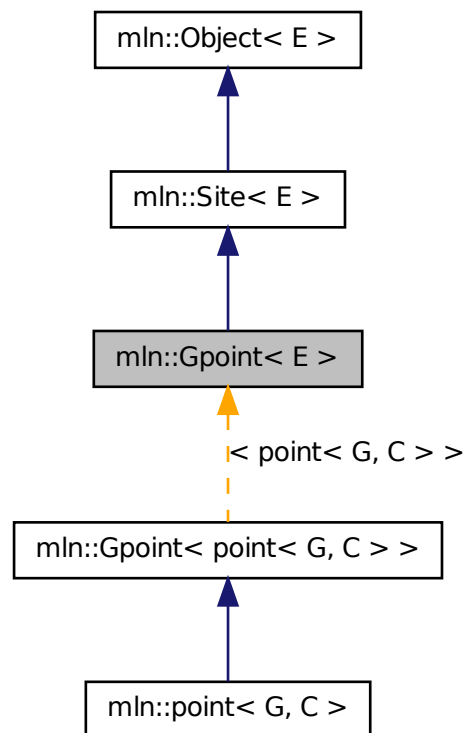
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

10.194 mln::Gpoint< E > Struct Template Reference

Base class for implementation of point classes.

`#include <gpoint.hh>`

Inheritance diagram for `mln::Gpoint< E >`:



Related Functions

(Note that these are not member functions.)

- template<typename P , typename D >
P **operator+** (const Gpoint< P > &p, const Gdpoint< D > &dp)
Add a delta-point rhs to a grid point lhs.
- template<typename P , typename D >
P & **operator+=** (Gpoint< P > &p, const Gdpoint< D > &dp)
Shift a point by a delta-point dp.
- template<typename L , typename R >
L::delta **operator-** (const Gpoint< L > &lhs, const Gpoint< R > &rhs)
Difference between a couple of grid point lhs and rhs.
- template<typename P , typename D >
P & **operator-=** (Gpoint< P > &p, const Gdpoint< D > &dp)
Shift a point by the negate of a delta-point dp.
- template<typename P , typename D >
P **operator/** (const Gpoint< P > &p, const value::scalar_< D > &dp)
Divide a point by a scalar s.
- template<typename P >
std::ostream & **operator<<** (std::ostream &ostr, const Gpoint< P > &p)
Print a grid point p into the output stream ostr.
- template<typename L , typename R >
bool **operator==** (const Gpoint< L > &lhs, const Gpoint< R > &rhs)
Equality comparison between a couple of grid point lhs and rhs.

10.194.1 Detailed Description

template<typename E> struct mln::Gpoint< E >

Base class for implementation of point classes. A point is an element of a space.

For instance, mln::point2d is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 115 of file gpoint.hh.

10.194.2 Friends And Related Function Documentation

10.194.2.1 template<typename P , typename D > P **operator+** (const Gpoint< P > & p, const Gdpoint< D > & dp) [**related**]

Add a delta-point rhs to a grid point lhs.

Parameters

- [in] **p** A grid point.
- [in] **dp** A delta-point.

The type of dp has to compatible with the type of p.

Returns

A point (temporary object).

See also

[mln::Gdpoint](#)

Definition at line 385 of file gpoint.hh.

10.194.2.2 `template<typename P , typename D > P & operator+=(Gpoint< P > & p, const Gdpoint< D > & dp)` [**related**]

Shift a point by a delta-point `dp`.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point `p` once translated by `dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 428 of file gpoint.hh.

10.194.2.3 `template<typename L , typename R > L::delta operator- (const Gpoint< L > & lhs, const Gpoint< R > & rhs)` [**related**]

Difference between a couple of grid point `lhs` and `rhs`.

Parameters

[in] *lhs* A first grid point.

[in] *rhs* A second grid point.

Warning

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

Postcondition

The result, `dp`, is such as `lhs == rhs + dp`.

Returns

A delta point (temporary object).

See also

[mln::Gdpoint](#)

Definition at line 374 of file gpoint.hh.

10.194.2.4 `template<typename P , typename D > P & operator-= (Gpoint< P > & p, const Gdpoint< D > & dp) [related]`

Shift a point by the negate of a delta-point dp.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point p once translated by - dp.

Precondition

The type of dp has to be compatible with the type of p.

Definition at line 436 of file gpoint.hh.

10.194.2.5 `template<typename P , typename D > P operator/ (const Gpoint< P > & p, const value::scalar_< D > & dp) [related]`

Divide a point by a scalar s.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A scalar.

Returns

A reference to the point p once divided by s.

10.194.2.6 `template<typename P > std::ostream & operator<< (std::ostream & ostr, const Gpoint< P > & p) [related]`

Print a grid point p into the output stream ostr.

Parameters

[in, out] *ostr* An output stream.

[in] *p* A grid point.

Returns

The modified output stream ostr.

Definition at line 417 of file gpoint.hh.

References mln::debug::format().

10.194.2.7 `template<typename L , typename R > bool operator==(const Gpoint< L > & lhs, const Gpoint< R > & rhs)` **[related]**

Equality comparison between a couple of grid point `lhs` and `rhs`.

Parameters

[in] *lhs* A first grid point.
[in] *rhs* A second grid point.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

Returns

True if both grid points have the same coordinates, otherwise false.

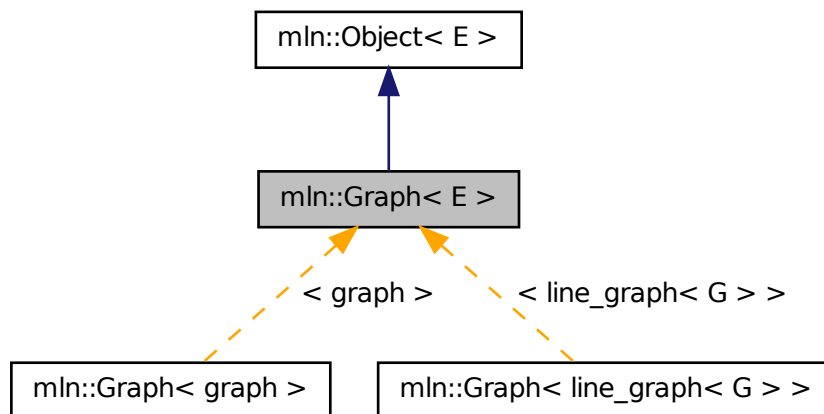
Definition at line 365 of file `gpoint.hh`.

10.195 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

```
#include <graph.hh>
```

Inheritance diagram for `mln::Graph< E >`:



10.195.1 Detailed Description

`template<typename E> struct mln::Graph< E >`

Base class for implementation of graph classes.

See also

mln::doc::Graph for a complete documentation of this class contents.

Definition at line 57 of file mln/core/concept/graph.hh.

10.196 mln::graph::attribute::card_t Struct Reference

Compute the cardinality of every component in a graph.

```
#include <card.hh>
```

Public Types

- typedef [util::array](#)< unsigned > [result](#)

Type of the computed value.

10.196.1 Detailed Description

Compute the cardinality of every component in a graph.

Returns

An array with the cardinality for each component. Components are labeled from 0.

Definition at line 61 of file graph/attribute/card.hh.

10.196.2 Member Typedef Documentation

10.196.2.1 typedef util::array<unsigned> mln::graph::attribute::card_t::result

Type of the computed value.

Definition at line 64 of file graph/attribute/card.hh.

10.197 mln::graph::attribute::representative_t Struct Reference

Compute the representative vertex of every component in a graph.

```
#include <representative.hh>
```

Public Types

- typedef [util::array](#)< unsigned > [result](#)

Type of the computed value.

10.197.1 Detailed Description

Compute the representative vertex of every component in a graph.

Returns

An array with the representative for each component. Components are labeled from 0.

Definition at line 63 of file representative.hh.

10.197.2 Member Typedef Documentation

10.197.2.1 `typedef util::array<unsigned> mln::graph::attribute::representative_t::result`

Type of the computed value.

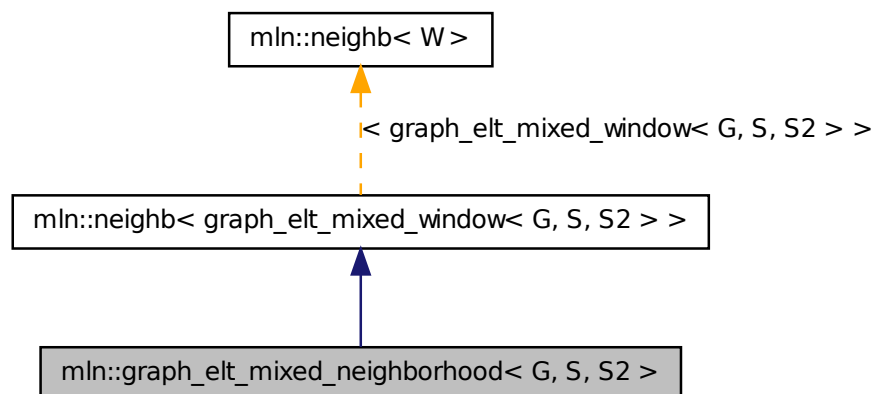
Definition at line 66 of file representative.hh.

10.198 `mln::graph_elt_mixed_neighborhood< G, S, S2 >` Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_mixed_neighborhood.hh>
```

Inheritance diagram for `mln::graph_elt_mixed_neighborhood< G, S, S2 >`:



Public Types

- `typedef neighb_bkd_niter< graph_elt_mixed_window< G, S, S2 > > bkd_niter`
Backward site iterator associated type.

- typedef neighb_fwd_niter< [graph_elt_mixed_window](#)< G, S, S2 > > [fwd_niter](#)
Forward site iterator associated type.
- typedef [fwd_niter](#) niter
Site iterator associated type.

10.198.1 Detailed Description

template<typename G, typename S, typename S2> struct mln::graph_elt_mixed_neighborhood< G, S, S2 >

Elementary neighborhood on graph class.

Template Parameters

- G* is a graph type.
- S* is a site set type.
- S2* is the site set type of the neighbors.

Definition at line 48 of file graph_elt_mixed_neighborhood.hh.

10.198.2 Member Typedef Documentation

10.198.2.1 typedef neighb_bkd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb< graph_elt_mixed_window< G, S, S2 > >::bkd_niter [\[inherited\]](#)

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.198.2.2 typedef neighb_fwd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb< graph_elt_mixed_window< G, S, S2 > >::fwd_niter [\[inherited\]](#)

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.198.2.3 typedef fwd_niter mln::neighb< graph_elt_mixed_window< G, S, S2 > >::niter [\[inherited\]](#)

Site iterator associated type.

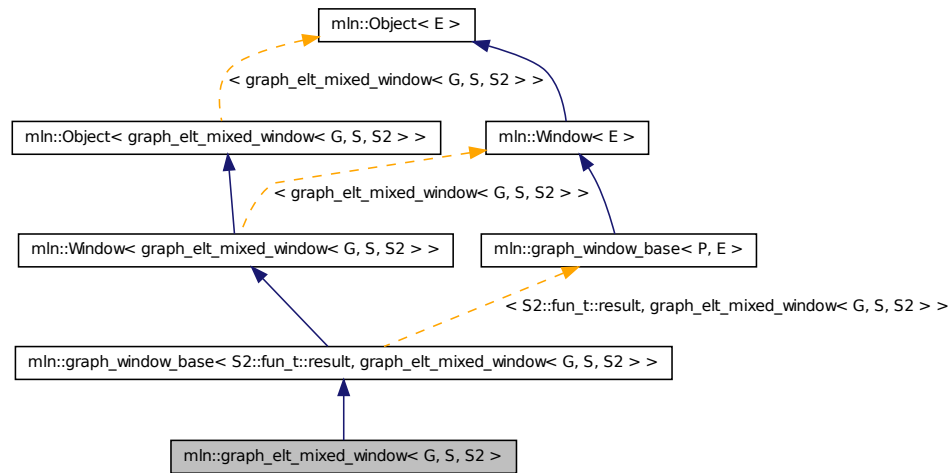
Definition at line 90 of file mln/core/neighb.hh.

10.199 mln::graph_elt_mixed_window< G, S, S2 > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_mixed_window.hh>
```

Inheritance diagram for mln::graph_elt_mixed_window< G, S, S2 >:



Public Types

- typedef `super_::target` [target](#)
Associated types.
- typedef `target::psite` [psite](#)
The type of psite corresponding to the window.
- typedef `S::psite` [center_t](#)
Type of the window center element.
- typedef `target::graph_element` [graph_element](#)
Type of the graph element pointed by this iterator.
- typedef `graph_window_piter< target, self_, nbh_fwd_iter_ >` [fwd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `graph_window_piter< target, self_, nbh_bkd_iter_ >` [bkd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `fwd_qiter` [qiter](#)
The default qiter type.
- typedef `S2::fun_t::result` [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.199.1 Detailed Description

template<typename G, typename S, typename S2> class mln::graph_elt_mixed_window< G, S, S2 >

Elementary window on graph class. G is the graph type. S is an image site set from where the center is extracted. S2 is an image site set from where the neighbors are extracted.

Definition at line 109 of file graph_elt_mixed_window.hh.

10.199.2 Member Typedef Documentation

10.199.2.1 template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self,nbh_bkd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::bkd_qiter

[Site_Iterator](#) type to browse the psites of the window w.r.t. the reverse ordering of vertices.

Definition at line 139 of file graph_elt_mixed_window.hh.

10.199.2.2 template<typename G , typename S , typename S2 > typedef S ::psite mln::graph_elt_mixed_window< G, S, S2 >::center_t

Type of the window center element.

Definition at line 128 of file graph_elt_mixed_window.hh.

10.199.2.3 `template<typename G , typename S , typename S2 > typedef
graph_window_piter<target,self,nbh_fwd_iter_> mln::graph_elt_mixed_window<
G, S, S2 >::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t.
the ordering of vertices.

Definition at line 135 of file `graph_elt_mixed_window.hh`.

10.199.2.4 `template<typename G , typename S , typename S2 > typedef target ::graph_element
mln::graph_elt_mixed_window< G, S, S2 >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 131 of file `graph_elt_mixed_window.hh`.

10.199.2.5 `template<typename G , typename S , typename S2 > typedef target ::psite
mln::graph_elt_mixed_window< G, S, S2 >::psite`

The type of psite corresponding to the window.

Definition at line 125 of file `graph_elt_mixed_window.hh`.

10.199.2.6 `template<typename G , typename S , typename S2 > typedef fwd_qiter
mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

Definition at line 142 of file `graph_elt_mixed_window.hh`.

10.199.2.7 `typedef S2::fun_t::result mln::graph_window_base< S2::fun_t::result ,
graph_elt_mixed_window< G, S, S2 > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

10.199.2.8 `template<typename G , typename S , typename S2 > typedef super_::target
mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

Definition at line 123 of file `graph_elt_mixed_window.hh`.

10.199.3 Member Function Documentation

10.199.3.1 `unsigned mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G,
S, S2 > >::delta () const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.199.3.2 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_centered () const [inherited]`

Is the window centered?

10.199.3.3 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_empty () const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.199.3.4 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_symmetric () const [inherited]`

Is the window symmetric?

10.199.3.5 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_valid () const [inherited]`

Return true by default.

10.199.3.6 `self_& mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::sym () [inherited]`

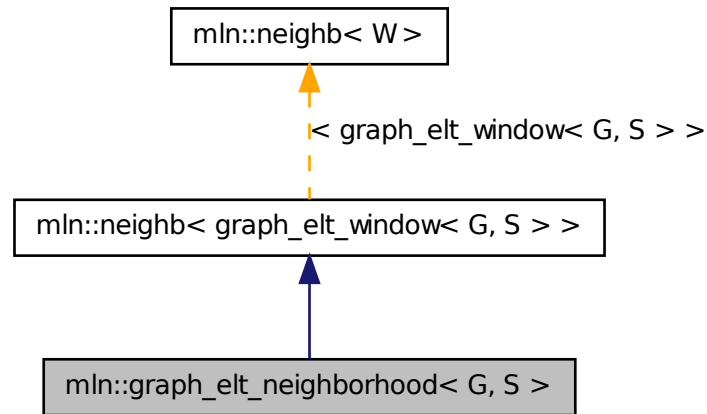
Apply a central symmetry to the target window.

10.200 mln::graph_elt_neighborhood< G, S > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_neighborhood.hh>
```

Inheritance diagram for `mln::graph_elt_neighborhood< G, S >`:



Public Types

- `typedef neighb_bkd_niter< graph_elt_window< G, S > > bkd_niter`
Backward site iterator associated type.
- `typedef neighb_fwd_niter< graph_elt_window< G, S > > fwd_niter`
Forward site iterator associated type.
- `typedef fwd_niter niter`
Site iterator associated type.

10.200.1 Detailed Description

`template<typename G, typename S> struct mln::graph_elt_neighborhood< G, S >`

Elementary neighborhood on graph class.

Template Parameters

G is a graph type.

S is a site set type.

Definition at line 47 of file `graph_elt_neighborhood.hh`.

10.200.2 Member Typedef Documentation

10.200.2.1 `typedef neighb_bkd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::bkd_niter [inherited]`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.200.2.2 `typedef neighb_fwd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::fwd_niter [inherited]`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.200.2.3 `typedef fwd_niter mln::neighb< graph_elt_window< G, S > >::niter [inherited]`

Site iterator associated type.

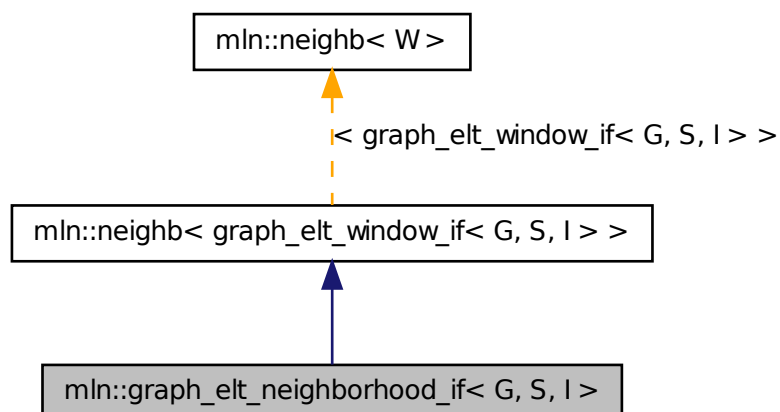
Definition at line 90 of file mln/core/neighb.hh.

10.201 mln::graph_elt_neighborhood_if< G, S, I > Struct Template Reference

Elementary neighborhood_if on graph class.

```
#include <graph_elt_neighborhood_if.hh>
```

Inheritance diagram for mln::graph_elt_neighborhood_if< G, S, I >:



Public Types

- typedef neighb_bkd_niter< [graph_elt_window_if](#)< G, S, I > > [bkd_niter](#)
Backward site iterator associated type.
- typedef neighb_fwd_niter< [graph_elt_window_if](#)< G, S, I > > [fwd_niter](#)
Forward site iterator associated type.
- typedef [fwd_niter](#) niter
Site iterator associated type.

Public Member Functions

- [graph_elt_neighborhood_if](#) ()
Constructors @ { Construct an invalid neighborhood.
- [graph_elt_neighborhood_if](#) (const [Image](#)< I > &mask)
- const I & [mask](#) () const
@ }

10.201.1 Detailed Description

template<typename G, typename S, typename I> struct mln::graph_elt_neighborhood_if< G, S, I >

Elementary neighborhood_if on graph class.

Definition at line 43 of file graph_elt_neighborhood_if.hh.

10.201.2 Member Typedef Documentation

10.201.2.1 typedef neighb_bkd_niter<graph_elt_window_if< G, S, I > > mln::neighb<graph_elt_window_if< G, S, I > >::bkd_niter [\[inherited\]](#)

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.201.2.2 typedef neighb_fwd_niter<graph_elt_window_if< G, S, I > > mln::neighb<graph_elt_window_if< G, S, I > >::fwd_niter [\[inherited\]](#)

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.201.2.3 `typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter`
`[inherited]`

Site iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.201.3 Constructor & Destructor Documentation

10.201.3.1 `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`
`G, S, I >::graph_elt_neighborhood_if() [inline]`

Constructors @ { Construct an invalid neighborhood.

Definition at line 67 of file graph_elt_neighborhood_if.hh.

10.201.3.2 `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`
`G, S, I >::graph_elt_neighborhood_if(const Image< I > & mask) [inline]`

Parameters

[in] *mask* A graph image of Boolean.

Definition at line 74 of file graph_elt_neighborhood_if.hh.

10.201.4 Member Function Documentation

10.201.4.1 `template<typename G , typename S , typename I > const I &`
`mln::graph_elt_neighborhood_if< G, S, I >::mask() const [inline]`

@ {

Return the graph image used as mask.

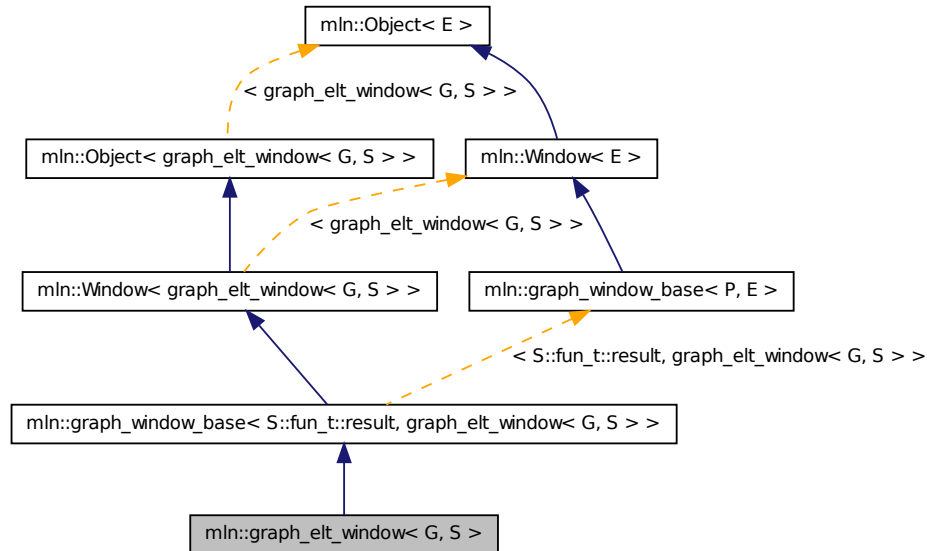
Definition at line 83 of file graph_elt_neighborhood_if.hh.

10.202 mln::graph_elt_window< G, S > Class Template Reference

Elementary window on graph class.

`#include <graph_elt_window.hh>`

Inheritance diagram for `mln::graph_elt_window< G, S >`:



Public Types

- typedef `S` [target](#)
Associated types.
- typedef `S::psite` [psite](#)
The type of psite corresponding to the window.
- typedef `S::psite` [center_t](#)
Type of the window center element.
- typedef `S::graph_element` [graph_element](#)
Type of the graph element pointed by this iterator.
- typedef [graph_window_piter](#)`< S, self_, nbh_fwd_iter_ >` [fwd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef [graph_window_piter](#)`< S, self_, nbh_bkd_iter_ >` [bkd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef `S::fun_t::result` [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_](#) & [sym](#) ()
Apply a central symmetry to the target window.

10.202.1 Detailed Description

template<typename G, typename S> class mln::graph_elt_window< G, S >

Elementary window on graph class. *G* is the graph type. *S* is an image site set from where the center is extracted. *S2* is an image site set from where the neighbors are extracted.

Definition at line 111 of file graph_elt_window.hh.

10.202.2 Member Typedef Documentation

10.202.2.1 template<typename G , typename S > typedef graph_window_iterator<S,self_nbh_bkd_iter_> mln::graph_elt_window< G, S >::bkd_qiter

[Site_Iterator](#) type to browse the psites of the window w.r.t.
the reverse ordering of vertices.

Definition at line 142 of file graph_elt_window.hh.

10.202.2.2 template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::center_t

Type of the window center element.

Definition at line 131 of file graph_elt_window.hh.

10.202.2.3 `template<typename G , typename S > typedef graph_window_
piter<S,self,nbh_fwd_iter_> mln::graph_elt_window< G, S
>::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

Definition at line 138 of file graph_elt_window.hh.

10.202.2.4 `template<typename G , typename S > typedef S ::graph_element
mln::graph_elt_window< G, S >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 134 of file graph_elt_window.hh.

10.202.2.5 `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S
>::psite`

The type of psite corresponding to the window.

Definition at line 128 of file graph_elt_window.hh.

10.202.2.6 `template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G,
S >::qiter`

The default qiter type.

Definition at line 145 of file graph_elt_window.hh.

10.202.2.7 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result ,
graph_elt_window< G, S > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.202.2.8 `template<typename G , typename S > typedef S mln::graph_elt_window< G, S
>::target`

Associated types.

Definition at line 125 of file graph_elt_window.hh.

10.202.3 Member Function Documentation

10.202.3.1 `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::delta () const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.202.3.2 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_centered () const [inherited]`

Is the window centered?

10.202.3.3 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_empty () const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.202.3.4 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_symmetric () const [inherited]`

Is the window symmetric?

10.202.3.5 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_valid () const [inherited]`

Return true by default.

10.202.3.6 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::sym () [inherited]`

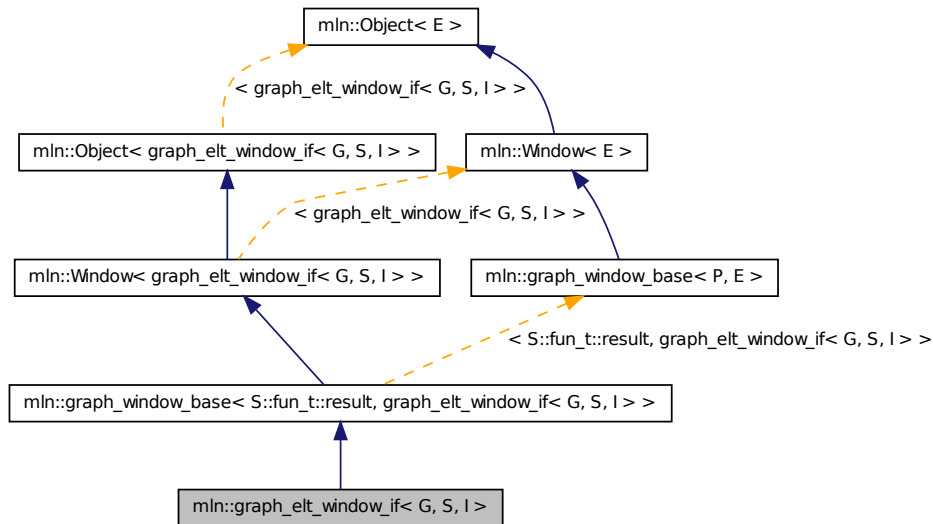
Apply a central symmetry to the target window.

10.203 mln::graph_elt_window_if< G, S, I > Class Template Reference

Custom window on graph class.

```
#include <graph_elt_window_if.hh>
```

Inheritance diagram for `mln::graph_elt_window_if< G, S, I >`:



Public Types

- typedef I [mask_t](#)
The type of the image used as mask.
- typedef S [target](#)
@}
- typedef target::psite [psite](#)
The type of psite corresponding to the window.
- typedef [graph_window_if_piter< target, self_, nbh_fwd_iter_ >](#) [fwd_qiter](#)
[Site_Iterator](#) type to browse the psites of the window w.r.t.
- typedef [graph_window_if_piter< target, self_, nbh_bkd_iter_ >](#) [bkd_qiter](#)
[Site_Iterator](#) type to browse the psites of the window w.r.t.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef S::fun_t::result [site](#)
Associated types.

Public Member Functions

- void [change_mask](#) (const [Image](#)< I > &mask)
Change mask image.
- [graph_elt_window_if](#) ()
Constructor.
- [graph_elt_window_if](#) (const [Image](#)< I > &mask)
- bool [is_valid](#) () const
Return true by default.
- const I & [mask](#) () const
Return the graph image used as mask.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_](#) & [sym](#) ()
Apply a central symmetry to the target window.

10.203.1 Detailed Description

template<typename G, typename S, typename I> class mln::graph_elt_window_if< G, S, I >

Custom window on graph class. It is defined thanks to a mask.

G is the graph type. S is the image site set. I is the graph image the type used as mask.

Definition at line 105 of file graph_elt_window_if.hh.

10.203.2 Member Typedef Documentation

10.203.2.1 template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self_nbh_bkd_iter_> mln::graph_elt_window_if< G, S, I >::bkd_qiter

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 147 of file graph_elt_window_if.hh.

10.203.2.2 `template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self,nbh_fwd_iter_> mln::graph_elt_window_if< G, S, I >::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

Definition at line 143 of file `graph_elt_window_if.hh`.

10.203.2.3 `template<typename G , typename S , typename I > typedef I mln::graph_elt_window_if< G, S, I >::mask_t`

The type of the image used as mask.

Definition at line 119 of file `graph_elt_window_if.hh`.

10.203.2.4 `template<typename G , typename S , typename I > typedef target::psite mln::graph_elt_window_if< G, S, I >::psite`

The type of psite corresponding to the window.

Definition at line 139 of file `graph_elt_window_if.hh`.

10.203.2.5 `template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

Definition at line 150 of file `graph_elt_window_if.hh`.

10.203.2.6 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

10.203.2.7 `template<typename G , typename S , typename I > typedef S mln::graph_elt_window_if< G, S, I >::target`

@ }

Associated types. The image domain on which this window iterates on.

Definition at line 136 of file `graph_elt_window_if.hh`.

10.203.3 Constructor & Destructor Documentation

10.203.3.1 `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if() [inline]`

Constructor.

@{ Default. Construct an invalid window.

Definition at line 174 of file graph_elt_window_if.hh.

10.203.3.2 `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if(const Image< I > & mask) [inline]`

Parameters

[in] *mask* A graph image of bool.

See also

[vertex_image](#), [edge_image](#).

Definition at line 181 of file graph_elt_window_if.hh.

10.203.4 Member Function Documentation

10.203.4.1 `template<typename G , typename S , typename I > void mln::graph_elt_window_if< G, S, I >::change_mask(const Image< I > & mask) [inline]`

Change mask image.

Definition at line 199 of file graph_elt_window_if.hh.

References `mln::graph_elt_window_if< G, S, I >::is_valid()`.

10.203.4.2 `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::delta() const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.203.4.3 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_centered() const [inherited]`

Is the window centered?

10.203.4.4 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_empty() const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.203.4.5 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I >
>::is_symmetric () const [inherited]`

Is the window symmetric?

10.203.4.6 `template<typename G , typename S , typename I > bool mln::graph_elt_window_if<
G, S, I >::is_valid () const [inline]`

Return true by default.

Reimplemented from [mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >](#).

Definition at line 208 of file `graph_elt_window_if.hh`.

Referenced by `mln::graph_elt_window_if< G, S, I >::change_mask()`.

10.203.4.7 `template<typename G , typename S , typename I > const I &
mln::graph_elt_window_if< G, S, I >::mask () const [inline]`

Return the graph image used as mask.

Definition at line 190 of file `graph_elt_window_if.hh`.

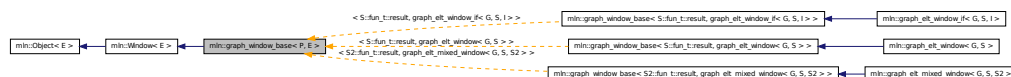
10.203.4.8 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I >
>::sym () [inherited]`

Apply a central symmetry to the target window.

10.204 mln::graph_window_base< P, E > Class Template Reference

`#include <graph_window_base.hh>`

Inheritance diagram for `mln::graph_window_base< P, E >`:



Public Types

- typedef [P](#) [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.

- bool [is_empty](#) () const
Interface of the concept [Window](#).
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.204.1 Detailed Description

template<typename P, typename E> class mln::graph_window_base< P, E >

Template Parameters

P [Site](#) type.

Definition at line 40 of file graph_window_base.hh.

10.204.2 Member Typedef Documentation

10.204.2.1 template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.204.3 Member Function Documentation

10.204.3.1 template<typename P , typename E > unsigned mln::graph_window_base< P, E >::delta () const [inline]

Return the maximum coordinate gap between the window center and a window point.

Definition at line 128 of file graph_window_base.hh.

10.204.3.2 template<typename P , typename E > bool mln::graph_window_base< P, E >::is_centered () const [inline]

Is the window centered?

Definition at line 112 of file graph_window_base.hh.

10.204.3.3 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_empty () const [inline]`

Interface of the concept [Window](#).

Is the window is empty?

Definition at line 104 of file graph_window_base.hh.

10.204.3.4 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric () const [inline]`

Is the window symmetric?

Definition at line 120 of file graph_window_base.hh.

10.204.3.5 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_valid () const [inline]`

Return true by default.

Reimplemented in [mln::graph_elt_window_if< G, S, I >](#).

Definition at line 153 of file graph_window_base.hh.

10.204.3.6 `template<typename P , typename E > graph_window_base< P, E > & mln::graph_window_base< P, E >::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 137 of file graph_window_base.hh.

10.205 mln::graph_window_if_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_if_piter.hh>
```

Inherits [site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >](#), and [is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >](#).

Public Types

- `typedef S::fun_t::result P`
Associated types.

Public Member Functions

- `void next ()`
Go to the next element.

- [graph_window_if_piter](#) ()
Construction.
- const S::graph_element & [element](#) () const
Return the graph element pointed by this iterator.
- unsigned [id](#) () const
Return the graph element id.

10.205.1 Detailed Description

template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >

Forward iterator on line graph window.

Definition at line 47 of file graph_window_if_piter.hh.

10.205.2 Member Typedef Documentation

**10.205.2.1 template<typename S , typename W , typename I > typedef S::fun_t ::result
 mln::graph_window_if_piter< S, W, I >::P**

Associated types.

Definition at line 60 of file graph_window_if_piter.hh.

10.205.3 Constructor & Destructor Documentation

**10.205.3.1 template<typename S , typename W , typename I > mln::graph_window_if_piter< S,
 W, I >::graph_window_if_piter () [inline]**

Construction.

Definition at line 122 of file graph_window_if_piter.hh.

10.205.4 Member Function Documentation

**10.205.4.1 template<typename S , typename W , typename I > const S::graph_element &
 mln::graph_window_if_piter< S, W, I >::element () const [inline]**

Return the graph element pointed by this iterator.

Definition at line 213 of file graph_window_if_piter.hh.

**10.205.4.2 template<typename S , typename W , typename I > unsigned
 mln::graph_window_if_piter< S, W, I >::id () const [inline]**

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 221 of file graph_window_if_piter.hh.

10.205.4.3 `void mln::Site_Iterator< graph_window_if_piter< S, W, I > >::next ()`
[inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.206 mln::graph_window_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >`, and `impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >`.

Public Types

- typedef `S::fun_t::result` [P](#)
Associated types
Type of the window elements.
- typedef `W::center_t` [center_t](#)
Type of the window center.
- typedef `W::graph_element` [graph_element](#)
Type of the graph element pointed by this iterator.

Public Member Functions

- void [change_target_site_set](#) (const S &s)
Change the target site set.
- void [next](#) ()
Go to the next element.
- const S & [target_site_set](#) () const

Return the target site set.

- [graph_window_piter](#) ()
Construction.
- `template<typename Pref >`
`graph_window_piter (const Window< W > &win, const Pref &p_ref)`
To be used in case the center and neighbor sites have the same type and belong to the same site set.
- `template<typename Pref >`
`graph_window_piter (const Window< W > &win, const Site_Set< S > &target_site_set, const Pref &p_ref)`
To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.
- `const graph_element & element () const`
Return the graph element pointed by this iterator.
- `unsigned id () const`
Return the graph element id.

10.206.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_piter< S, W, I >`

Forward iterator on line graph window.

Template Parameters

S is the site set type.

W is the window type.

I is the underlying iterator type.

Definition at line 99 of file `graph_window_piter.hh`.

10.206.2 Member Typedef Documentation

10.206.2.1 `template<typename S , typename W , typename I > typedef W::center_t
mln::graph_window_piter< S, W, I >::center_t`

Type of the window center.

Definition at line 120 of file `graph_window_piter.hh`.

10.206.2.2 `template<typename S , typename W , typename I > typedef W::graph_element
mln::graph_window_piter< S, W, I >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 122 of file `graph_window_piter.hh`.

10.206.2.3 `template<typename S , typename W , typename I > typedef S::fun_t ::result
mln::graph_window_piter< S, W, I >::P`

Associated types

Type of the window elements.

Definition at line 118 of file graph_window_piter.hh.

10.206.3 Constructor & Destructor Documentation

10.206.3.1 `template<typename S , typename W , typename I > mln::graph_window_piter< S, W,
I >::graph_window_piter () [inline]`

Construction.

Definition at line 226 of file graph_window_piter.hh.

10.206.3.2 `template<typename S , typename W , typename I > template<typename Pref >
mln::graph_window_piter< S, W, I >::graph_window_piter (const Window< W > &
win, const Pref & p_ref) [inline]`

To be used in case the center and neighbor sites have the same type and belong to the same site set.

Parameters

win The underlying window.

p_ref [Window](#) center.

Definition at line 235 of file graph_window_piter.hh.

10.206.3.3 `template<typename S , typename W , typename I > template<typename Pref >
mln::graph_window_piter< S, W, I >::graph_window_piter (const Window< W > &
win, const Site_Set< S > & target_site_set, const Pref & p_ref) [inline]`

To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.

Parameters

win The underlying window.

target_site_set [Site](#) set in which neighbor sites are extracted.

p_ref [Window](#) center.

Definition at line 249 of file graph_window_piter.hh.

10.206.4 Member Function Documentation

10.206.4.1 `template<typename S , typename W , typename I > void mln::graph_window_piter<
S, W, I >::change_target_site_set (const S & s) [inline]`

Change the target site set.

[Window](#) elements different from the center come from the target site set.

Definition at line 357 of file graph_window_piter.hh.

10.206.4.2 `template<typename S , typename W , typename I > const graph_window_piter< S, W, I >::graph_element & mln::graph_window_piter< S, W, I >::element () const [inline]`

Return the graph element pointed by this iterator.

Definition at line 341 of file graph_window_piter.hh.

10.206.4.3 `template<typename S , typename W , typename I > unsigned mln::graph_window_piter< S, W, I >::id () const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 349 of file graph_window_piter.hh.

10.206.4.4 `void mln::Site_Iterator< graph_window_piter< S, W, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.206.4.5 `template<typename S , typename W , typename I > const S & mln::graph_window_piter< S, W, I >::target_site_set () const [inline]`

Return the target site set.

[Window](#) elements different from the center come from the target site set.

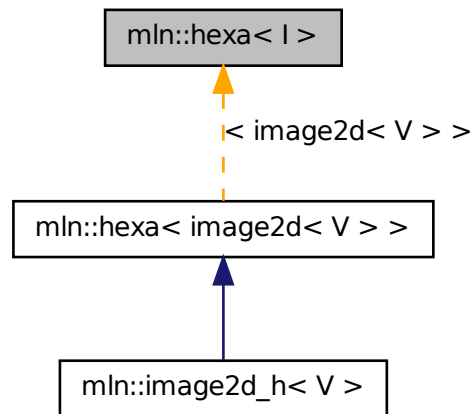
Definition at line 366 of file graph_window_piter.hh.

10.207 mln::hexa< I > Struct Template Reference

hexagonal image class.

```
#include <hexa.hh>
```

Inheritance diagram for `mln::hexa< I >`:



Public Types

- typedef `hexa_bkd_piter_< box2d > bkd_piter`
FIXME : should it be in box2d_h? Backward [Site_Iterator](#) associated type.
- typedef `hexa_fwd_piter_< box2d > fwd_piter`
FIXME : should it be in box2d_h? Forward [Site_Iterator](#) associated type.
- typedef `I::lvalue lvalue`
Lvalue associated type.
- typedef `point2d_h psite`
[Point](#) site type.
- typedef `I::rvalue rvalue`
Return type of read-only access.
- typedef `hexa< tag::image_< I > > skeleton`
Skeleton.
- typedef `I::value value`
[Value](#) associated type.

Public Member Functions

- const `box2d_h & domain ()` const

Give the definition domain.

- `bool has (const psite &p) const`
Test if p belongs to the image domain.
- `hexa (I &ima)`
Constructor with an base image.
- `hexa ()`
Constructor without argument.
- `lvalue operator\(\) (const point2d_h &p)`
Read-write access of pixel value at hexa point site p .
- `rvalue operator\(\) (const point2d_h &p) const`
Read-only access of pixel value at hexa point site p .

10.207.1 Detailed Description

`template<typename I> struct mln::hexa< I >`

hexagonal image class. The parameter I is the type of the base image. This image class which handles hexagonal grid.

```
Ex : 1 3 5 7 9 11 0 2 4 6 8 10 ----- 0 XX| | | | |XX ----- 2 XX| | | | |XX
----- 4 XX| | | | |XX ----- 6 XX| | | | |XX ----- 8 XX| | | | |
|XX -----
```

Definition at line 116 of file hexa.hh.

10.207.2 Member Typedef Documentation

10.207.2.1 `template<typename I> typedef hexa_bkd_piter_<box2d> mln::hexa< I >::bkd_piter`

FIXME : should it be in box2d_h? Backward [Site_Iterator](#) associated type.

Definition at line 140 of file hexa.hh.

10.207.2.2 `template<typename I> typedef hexa_fwd_piter_<box2d> mln::hexa< I >::fwd_piter`

FIXME : should it be in box2d_h? Forward [Site_Iterator](#) associated type.

Definition at line 136 of file hexa.hh.

10.207.2.3 `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue`

Lvalue associated type.

Definition at line 126 of file hexa.hh.

10.207.2.4 `template<typename I> typedef point2d_h mln::hexa< I >::psite`

[Point](#) site type.

Reimplemented in [mln::image2d_h< V >](#).

Definition at line 132 of file hexa.hh.

10.207.2.5 `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

Definition at line 129 of file hexa.hh.

10.207.2.6 `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

Definition at line 120 of file hexa.hh.

10.207.2.7 `template<typename I> typedef I ::value mln::hexa< I >::value`

[Value](#) associated type.

Definition at line 123 of file hexa.hh.

10.207.3 Constructor & Destructor Documentation**10.207.3.1 `template<typename I> mln::hexa< I >::hexa () [inline]`**

Constructor without argument.

Definition at line 215 of file hexa.hh.

10.207.3.2 `template<typename I> mln::hexa< I >::hexa (I & ima) [inline]`

Constructor with an base image.

Definition at line 222 of file hexa.hh.

10.207.4 Member Function Documentation**10.207.4.1 `template<typename I> const box2d_h & mln::hexa< I >::domain () const [inline]`**

Give the definition domain.

Definition at line 250 of file hexa.hh.

10.207.4.2 `template<typename I> bool mln::hexa< I >::has (const psite & p) const [inline]`

Test if *p* belongs to the image domain.

Definition at line 259 of file hexa.hh.

Referenced by mln::hexa< I >::operator()().

10.207.4.3 `template<typename I> hexa< I >::rvalue mln::hexa< I >::operator() (const point2d_h & p) const [inline]`

Read-only access of pixel value at hexa point site p.

Definition at line 230 of file hexa.hh.

References mln::hexa< I >::has().

10.207.4.4 `template<typename I> hexa< I >::lvalue mln::hexa< I >::operator() (const point2d_h & p) [inline]`

Read-write access of pixel value at hexa point site p.

Definition at line 240 of file hexa.hh.

References mln::hexa< I >::has().

10.208 mln::histo::array< T > Struct Template Reference

Generic histogram class over a value set with type T.

```
#include <array.hh>
```

10.208.1 Detailed Description

`template<typename T> struct mln::histo::array< T >`

Generic histogram class over a value set with type T.

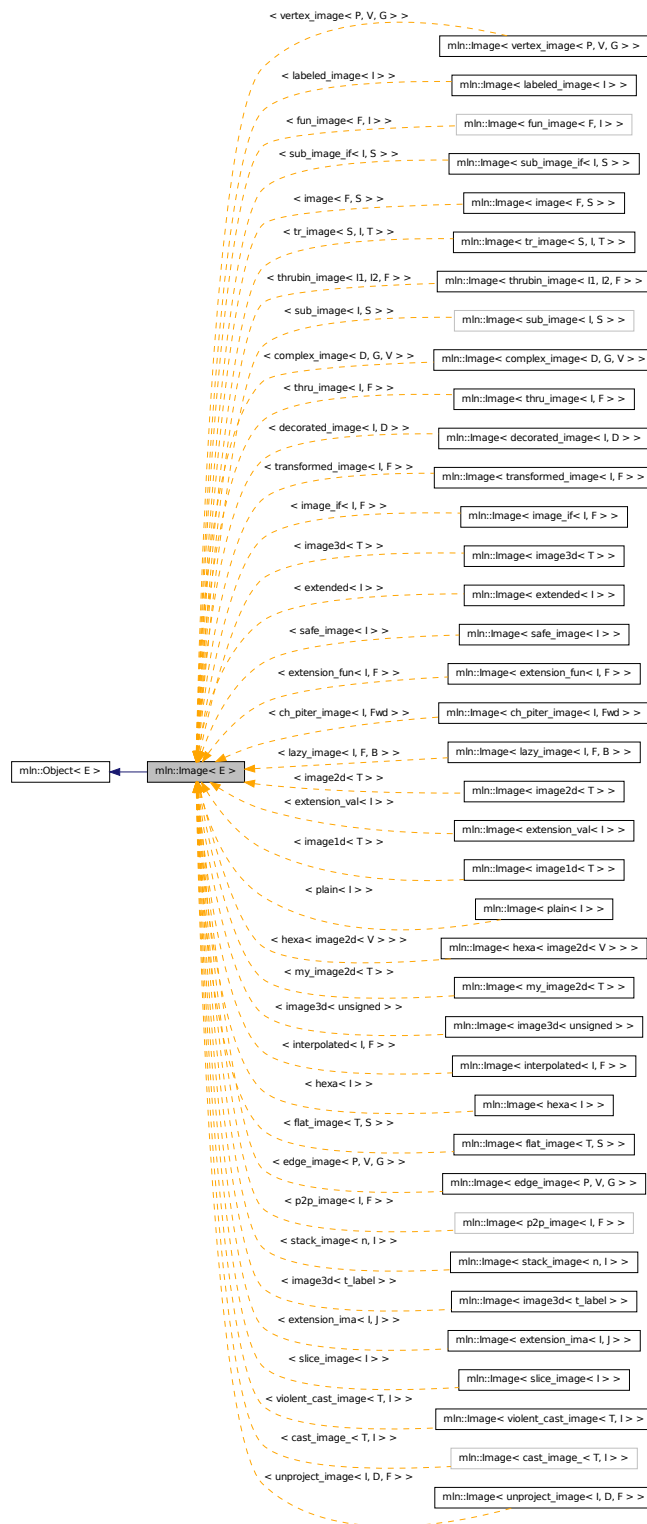
Definition at line 48 of file histo/array.hh.

10.209 mln::Image< E > Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```

Inheritance diagram for mln::Image< E >:



10.209.1 Detailed Description

`template<typename E> struct mln::Image< E >`

Base class for implementation of image classes.

See also

[mln::doc::Image](#) for a complete documentation of this class contents.

Definition at line 73 of file core/concept/image.hh.

10.210 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits `image_primary< T, box1d, image1d< T > >`.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image1d](#)< tag::value_< T > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const [box1d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint1d](#) &dp) const

Give the offset corresponding to the delta-point `dp`.

- `const box1d & domain () const`
Give the definition domain.
- `const T & element (unsigned i) const`
Read-only access to the `i-th` image value (including the border).
- `T & element (unsigned i)`
Read-write access to the `i-th` image value (including the border).
- `bool has (const point1d &p) const`
Test if `p` is valid.
- `image1d ()`
Constructor without argument.
- `image1d (const box1d &b, unsigned bdr=border::thickness)`
Constructor with a box and the border thickness.
- `image1d (unsigned ninds, unsigned bdr=border::thickness)`
Constructor with the number of indices and the border thickness.
- `unsigned nelements () const`
Give the number of cells (points including border ones).
- `unsigned ninds () const`
Give the number of indexes.
- `T & operator() (const point1d &p)`
Read-write access to the image value located at point `p`.
- `const T & operator() (const point1d &p) const`
Read-only access to the image value located at point `p`.
- `point1d point_at_index (unsigned i) const`
Give the point corresponding to the offset `o`.
- `const box1d & vbox () const`
virtual box, i.e., box including the virtual border

10.210.1 Detailed Description

```
template<typename T> struct mln::image1d< T >
```

Basic 1D image class. The parameter `T` is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness before and after data.

Definition at line 155 of file `image1d.hh`.

10.210.2 Member Typedef Documentation

10.210.2.1 `template<typename T> typedef T& mln::image1d< T >::lvalue`

Return type of read-write access.

Definition at line 167 of file image1d.hh.

10.210.2.2 `template<typename T> typedef const T& mln::image1d< T >::rvalue`

Return type of read-only access.

Definition at line 164 of file image1d.hh.

10.210.2.3 `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton`

Skeleton.

Definition at line 170 of file image1d.hh.

10.210.2.4 `template<typename T> typedef T mln::image1d< T >::value`

[Value](#) associated type.

Definition at line 161 of file image1d.hh.

10.210.3 Constructor & Destructor Documentation

10.210.3.1 `template<typename T> mln::image1d< T >::image1d () [inline]`

Constructor without argument.

Definition at line 371 of file image1d.hh.

10.210.3.2 `template<typename T> mln::image1d< T >::image1d (unsigned ninds, unsigned bdr = border::thickness) [inline]`

Constructor with the number of indices and the border thickness.

Definition at line 384 of file image1d.hh.

References `mln::make::box1d()`.

10.210.3.3 `template<typename T> mln::image1d< T >::image1d (const box1d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness.

Definition at line 377 of file image1d.hh.

10.210.4 Member Function Documentation

10.210.4.1 `template<typename T> const box1d & mln::image1d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 411 of file image1d.hh.

10.210.4.2 `template<typename T> unsigned mln::image1d< T >::border () const [inline]`

Give the border thickness.

Definition at line 429 of file image1d.hh.

10.210.4.3 `template<typename T> const T * mln::image1d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 520 of file image1d.hh.

10.210.4.4 `template<typename T> T * mln::image1d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 529 of file image1d.hh.

10.210.4.5 `template<typename T> int mln::image1d< T >::delta_index (const dpoint1d & dp) const [inline]`

Give the offset corresponding to the delta-point dp.

Definition at line 538 of file image1d.hh.

10.210.4.6 `template<typename T> const box1d & mln::image1d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 402 of file image1d.hh.

10.210.4.7 `template<typename T> const T & mln::image1d< T >::element (unsigned i) const [inline]`

Read-only access to the *i*-th image value (including the border).

Definition at line 502 of file image1d.hh.

References mln::image1d< T >::nelements().

10.210.4.8 **template<typename T > T & mln::image1d< T >::element (unsigned i)** **[inline]**

Read-write access to the *i*-th image value (including the border).

Definition at line 511 of file image1d.hh.

References mln::image1d< T >::nelements().

10.210.4.9 **template<typename T > bool mln::image1d< T >::has (const point1d & p) const** **[inline]**

Test if *p* is valid.

Definition at line 447 of file image1d.hh.

Referenced by mln::image1d< T >::operator()().

10.210.4.10 **template<typename T > unsigned mln::image1d< T >::nelements () const** **[inline]**

Give the number of cells (points including border ones).

Definition at line 438 of file image1d.hh.

Referenced by mln::image1d< T >::element(), and mln::image1d< T >::point_at_index().

10.210.4.11 **template<typename T > unsigned mln::image1d< T >::ninds () const** **[inline]**

Give the number of indexes.

Definition at line 483 of file image1d.hh.

10.210.4.12 **template<typename T > const T & mln::image1d< T >::operator() (const point1d & p) const** **[inline]**

Read-only access to the image value located at point *p*.

Definition at line 456 of file image1d.hh.

References mln::image1d< T >::has().

10.210.4.13 **template<typename T > T & mln::image1d< T >::operator() (const point1d & p)** **[inline]**

Read-write access to the image value located at point *p*.

Definition at line 465 of file image1d.hh.

References mln::image1d< T >::has().

10.210.4.14 **template<typename T > point1d mln::image1d< T >::point_at_index (unsigned i)** **const [inline]**

Give the point corresponding to the offset *o*.

Definition at line 548 of file image1d.hh.

References `mln::image1d< T >::nelements()`.

10.210.4.15 `template<typename T> const box1d & mln::image1d< T >::vbbox () const` [inline]

virtual box, i.e., box including the virtual border

Definition at line 420 of file image1d.hh.

10.211 `mln::image2d< T >` Class Template Reference

Basic 2D image class.

`#include <image2d.hh>`

Inherits `image_primary< T, mln::box2d, image2d< T > >`.

Public Types

- typedef `T &` `lvalue`
Return type of read-write access.
- typedef `const T &` `rvalue`
Return type of read-only access.
- typedef `image2d< tag::value_< T > >` `skeleton`
Skeleton.
- typedef `T` `value`
Value associated type.

Public Member Functions

- const `box2d &` `bbox` () const
Give the bounding box domain.
- unsigned `border` () const
Give the border thickness.
- const `T *` `buffer` () const
Give a hook to the value buffer.
- `T *` `buffer` ()
Give a hook to the value buffer.
- int `delta_index` (const `dpoint2d &dp`) const
Give the delta-index corresponding to the delta-point `dp`.

- const `box2d` & `domain` () const
Give the definition domain.
- T & `element` (unsigned i)
Read-write access to the image value located at index i.
- const T & `element` (unsigned i) const
Read-only access to the image value located at index i.
- bool `has` (const `point2d` &p) const
Test if p is valid.
- `image2d` (int nrows, int ncols, unsigned bdr=border::thickness)
Constructor with the numbers of rows and columns and the border thickness.
- `image2d` ()
Constructor without argument.
- `image2d` (const `box2d` &b, unsigned bdr=border::thickness)
Constructor with a box and the border thickness (default is 3).
- unsigned `ncols` () const
Give the number of columns.
- unsigned `nelements` () const
Give the number of elements (points including border ones).
- unsigned `nrows` () const
Give the number of rows.
- const T & `operator()` (const `point2d` &p) const
Read-only access to the image value located at point p.
- T & `operator()` (const `point2d` &p)
Read-write access to the image value located at point p.
- `point2d point_at_index` (unsigned i) const
Give the point corresponding to the index i.

10.211.1 Detailed Description

`template<typename T> class mln::image2d< T >`

Basic 2D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 136 of file core/image/image2d.hh.

10.211.2 Member Typedef Documentation

10.211.2.1 `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

Definition at line 148 of file core/image/image2d.hh.

10.211.2.2 `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

Definition at line 145 of file core/image/image2d.hh.

10.211.2.3 `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

Definition at line 152 of file core/image/image2d.hh.

10.211.2.4 `template<typename T> typedef T mln::image2d< T >::value`

[Value](#) associated type.

Definition at line 142 of file core/image/image2d.hh.

10.211.3 Constructor & Destructor Documentation

10.211.3.1 `template<typename T > mln::image2d< T >::image2d () [inline]`

Constructor without argument.

Definition at line 394 of file core/image/image2d.hh.

10.211.3.2 `template<typename T > mln::image2d< T >::image2d (int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of rows and columns and the border thickness.

Definition at line 400 of file core/image/image2d.hh.

References `mln::make::box2d()`.

10.211.3.3 `template<typename T > mln::image2d< T >::image2d (const box2d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 407 of file core/image/image2d.hh.

10.211.4 Member Function Documentation

10.211.4.1 `template<typename T> const box2d & mln::image2d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 433 of file core/image/image2d.hh.

10.211.4.2 `template<typename T> unsigned mln::image2d< T >::border () const [inline]`

Give the border thickness.

Definition at line 520 of file core/image/image2d.hh.

10.211.4.3 `template<typename T> const T * mln::image2d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 556 of file core/image/image2d.hh.

10.211.4.4 `template<typename T> T * mln::image2d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 565 of file core/image/image2d.hh.

10.211.4.5 `template<typename T> int mln::image2d< T >::delta_index (const dpoint2d & dp) const [inline]`

Give the delta-index corresponding to the delta-point dp.

Definition at line 574 of file core/image/image2d.hh.

10.211.4.6 `template<typename T> const box2d & mln::image2d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 424 of file core/image/image2d.hh.

Referenced by mln::morpho::line_gradient(), mln::make_debug_graph_image(), and mln::io::txt::save().

10.211.4.7 `template<typename T> const T & mln::image2d< T >::element (unsigned i) const [inline]`

Read-only access to the image value located at index i.

Definition at line 538 of file core/image/image2d.hh.

References mln::image2d< T >::nelements().

10.211.4.8 **template<typename T> T & mln::image2d< T >::element (unsigned i)** **[inline]**

Read-write access to the image value located at index *i*.

Definition at line 547 of file core/image/image2d.hh.

References mln::image2d< T >::nelements().

10.211.4.9 **template<typename T> bool mln::image2d< T >::has (const point2d & p) const** **[inline]**

Test if *p* is valid.

Definition at line 451 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::operator()(), and mln::debug::put_word().

10.211.4.10 **template<typename T> unsigned mln::image2d< T >::ncols () const** **[inline]**

Give the number of columns.

Definition at line 508 of file core/image/image2d.hh.

10.211.4.11 **template<typename T> unsigned mln::image2d< T >::nelements () const** **[inline]**

Give the number of elements (points including border ones).

Definition at line 529 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::element(), and mln::image2d< T >::point_at_index().

10.211.4.12 **template<typename T> unsigned mln::image2d< T >::nrows () const** **[inline]**

Give the number of rows.

Definition at line 499 of file core/image/image2d.hh.

10.211.4.13 **template<typename T> T & mln::image2d< T >::operator() (const point2d & p)** **[inline]**

Read-write access to the image value located at point *p*.

Definition at line 469 of file core/image/image2d.hh.

References mln::image2d< T >::has().

10.211.4.14 **template<typename T> const T & mln::image2d< T >::operator() (const point2d & p) const** **[inline]**

Read-only access to the image value located at point *p*.

Definition at line 460 of file core/image/image2d.hh.

References mln::image2d< T >::has().

10.211.4.15 `template<typename T> point2d mln::image2d< T >::point_at_index (unsigned i)
const [inline]`

Give the point corresponding to the index *i*.

Definition at line 584 of file core/image/image2d.hh.

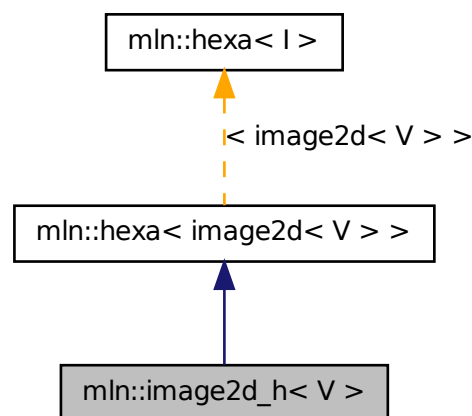
References `mln::image2d< T >::nelements()`.

10.212 mln::image2d_h< V > Struct Template Reference

2d image based on an hexagonal mesh.

`#include <image2d_h.hh>`

Inheritance diagram for `mln::image2d_h< V >`:



Public Types

- `typedef hexa_bkd_piter_< box2d > bkd_piter`
FIXME : should it be in box2d_h? Backward Site_Iterator associated type.
- `typedef hexa_fwd_piter_< box2d > fwd_piter`
FIXME : should it be in box2d_h? Forward Site_Iterator associated type.
- `typedef image2d< V >::lvalue lvalue`
Lvalue associated type.
- `typedef point2d_h psite`
Point site type.

- typedef `image2d< V >::rvalue rvalue`
Return type of read-only access.
- typedef `hexa< tag::image_< image2d< V > > > skeleton`
Skeleton.
- typedef `image2d< V >::value value`
Value associated type.

Public Member Functions

- const `box2d_h & domain ()` const
Give the definition domain.
- bool `has (const psite &p)` const
Test if p belongs to the image domain.
- `image2d_h (int nrow, int ncol, unsigned bdr=border::thickness)`
Constructor with the numbers of rows and columns border thickness.
- `lvalue operator() (const point2d_h &p)`
Read-write access of pixel value at hexa point site p .
- `rvalue operator() (const point2d_h &p)` const
Read-only access of pixel value at hexa point site p .

10.212.1 Detailed Description

`template<typename V> struct mln::image2d_h< V >`

2d image based on an hexagonal mesh.

Definition at line 50 of file `image2d_h.hh`.

10.212.2 Member Typedef Documentation

10.212.2.1 `typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter`
`[inherited]`

FIXME : should it be in `box2d_h`? Backward Site_Iterator associated type.

Definition at line 140 of file `hexa.hh`.

10.212.2.2 `typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter`
`[inherited]`

FIXME : should it be in `box2d_h`? Forward Site_Iterator associated type.

Definition at line 136 of file `hexa.hh`.

10.212.2.3 typedef image2d< V > ::lvalue mln::hexa< image2d< V > >::lvalue [inherited]

Lvalue associated type.

Definition at line 126 of file hexa.hh.

10.212.2.4 template<typename V > typedef point2d_h mln::image2d_h< V >::psite

[Point](#) site type.

Reimplemented from [mln::hexa< image2d< V > >](#).

Definition at line 56 of file image2d_h.hh.

10.212.2.5 typedef image2d< V > ::rvalue mln::hexa< image2d< V > >::rvalue [inherited]

Return type of read-only access.

Definition at line 129 of file hexa.hh.

10.212.2.6 typedef hexa< tag::image_<image2d< V > > > mln::hexa< image2d< V > >::skeleton [inherited]

Skeleton.

Definition at line 120 of file hexa.hh.

10.212.2.7 typedef image2d< V > ::value mln::hexa< image2d< V > >::value [inherited]

Value associated type.

Definition at line 123 of file hexa.hh.

10.212.3 Constructor & Destructor Documentation**10.212.3.1 template<typename V > mln::image2d_h< V >::image2d_h (int *nrows*, int *ncols*, unsigned *bdr* = *border::thickness*) [inline]**

Constructor with the numbers of rows and columns border thickness.

image2d_h(3,6) will build this hexa image :

```
1 3 5 0 2 4 ----- 0| x x x | 2| x x x | 4| x x x
```

Definition at line 82 of file image2d_h.hh.

10.212.4 Member Function Documentation**10.212.4.1 const box2d_h& mln::hexa< image2d< V > >::domain () const [inherited]**

Give the definition domain.

10.212.4.2 `bool mln::hexa< image2d< V > >::has (const psite & p) const` `[inherited]`

Test if `p` belongs to the image domain.

10.212.4.3 `rvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p) const` `[inherited]`

Read-only access of pixel value at hexa point site `p`.

10.212.4.4 `lvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p)` `[inherited]`

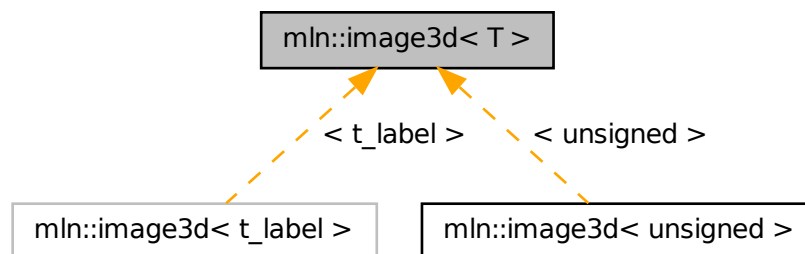
Read-write access of pixel value at hexa point site `p`.

10.213 `mln::image3d< T >` Struct Template Reference

Basic 3D image class.

```
#include <image3d.hh>
```

Inheritance diagram for `mln::image3d< T >`:



Public Types

- typedef `T` & `lvalue`
Return type of read-write access.
- typedef `const T` & `rvalue`
Return type of read-only access.
- typedef `image3d< tag::value_< T > >` `skeleton`
Skeleton.
- typedef `T` `value`

Value associated type.

Public Member Functions

- const [box3d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint3d](#) &dp) const
Fast [Image](#) method.
- const [box3d](#) & [domain](#) () const
Give the definition domain.
- const T & [element](#) (unsigned i) const
Read-only access to the image value located at index i .
- T & [element](#) (unsigned i)
Read-write access to the image value located at index i .
- bool [has](#) (const [point3d](#) &p) const
Test if p is valid.
- [image3d](#) (const [box3d](#) &b, unsigned bdr=[border::thickness](#))
Constructor with a box and the border thickness (default is 3).
- [image3d](#) ()
Constructor without argument.
- [image3d](#) (int nslis, int nrow, int ncol, unsigned bdr=[border::thickness](#))
Constructor with the numbers of indexes and the border thickness.
- unsigned [ncols](#) () const
Give the number of columns.
- unsigned [nelements](#) () const
Give the number of cells (points including border ones).
- unsigned [nrows](#) () const
Give the number of rows.

- unsigned `nslis` () const
Give the number of slices.
- const T & `operator()` (const `point3d` &p) const
Read-only access to the image value located at point p.
- T & `operator()` (const `point3d` &p)
Read-write access to the image value located at point p.
- `point3d point_at_index` (unsigned o) const
Give the point corresponding to the offset o.
- const `box3d` & `vbbox` () const
virtual box, i.e., box including the virtual border

10.213.1 Detailed Description

template<typename T> struct mln::image3d< T >

Basic 3D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 130 of file core/image/image3d.hh.

10.213.2 Member Typedef Documentation

10.213.2.1 template<typename T> typedef T& mln::image3d< T >::lvalue

Return type of read-write access.

Definition at line 153 of file core/image/image3d.hh.

10.213.2.2 template<typename T> typedef const T& mln::image3d< T >::rvalue

Return type of read-only access.

Definition at line 150 of file core/image/image3d.hh.

10.213.2.3 template<typename T> typedef image3d< tag::value_<T> > mln::image3d< T >::skeleton

Skeleton.

Definition at line 157 of file core/image/image3d.hh.

10.213.2.4 template<typename T> typedef T mln::image3d< T >::value

[Value](#) associated type.

Definition at line 147 of file core/image/image3d.hh.

10.213.3 Constructor & Destructor Documentation

10.213.3.1 `template<typename T> mln::image3d< T >::image3d () [inline]`

Constructor without argument.

Definition at line 389 of file core/image/image3d.hh.

10.213.3.2 `template<typename T> mln::image3d< T >::image3d (const box3d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 395 of file core/image/image3d.hh.

10.213.3.3 `template<typename T> mln::image3d< T >::image3d (int nslis, int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of indexes and the border thickness.

Definition at line 402 of file core/image/image3d.hh.

References mln::make::box3d().

10.213.4 Member Function Documentation

10.213.4.1 `template<typename T> const box3d & mln::image3d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 428 of file core/image/image3d.hh.

10.213.4.2 `template<typename T> unsigned mln::image3d< T >::border () const [inline]`

Give the border thickness.

Definition at line 446 of file core/image/image3d.hh.

10.213.4.3 `template<typename T> const T * mln::image3d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 554 of file core/image/image3d.hh.

10.213.4.4 `template<typename T> T * mln::image3d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 563 of file core/image/image3d.hh.

10.213.4.5 `template<typename T> int mln::image3d< T >::delta_index (const dpoint3d & dp) const [inline]`

Fast [Image](#) method.

Give the offset corresponding to the delta-point `dp`.

Definition at line 572 of file `core/image/image3d.hh`.

10.213.4.6 `template<typename T> const box3d & mln::image3d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 419 of file `core/image/image3d.hh`.

Referenced by `mln::accu::stat::operator==()`.

10.213.4.7 `template<typename T> const T & mln::image3d< T >::element (unsigned i) const [inline]`

Read-only access to the image value located at index `i`.

Definition at line 491 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

10.213.4.8 `template<typename T> T & mln::image3d< T >::element (unsigned i) [inline]`

Read-write access to the image value located at index `i`.

Definition at line 500 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

10.213.4.9 `template<typename T> bool mln::image3d< T >::has (const point3d & p) const [inline]`

Test if `p` is valid.

Definition at line 464 of file `core/image/image3d.hh`.

Referenced by `mln::image3d< T >::operator()()`.

10.213.4.10 `template<typename T> unsigned mln::image3d< T >::ncols () const [inline]`

Give the number of columns.

Definition at line 545 of file `core/image/image3d.hh`.

10.213.4.11 `template<typename T> unsigned mln::image3d< T >::nelements () const [inline]`

Give the number of cells (points including border ones).

Definition at line 455 of file core/image/image3d.hh.

Referenced by mln::image3d< T >::element(), and mln::image3d< T >::point_at_index().

10.213.4.12 `template<typename T> unsigned mln::image3d< T >::nrows () const [inline]`

Give the number of rows.

Definition at line 536 of file core/image/image3d.hh.

10.213.4.13 `template<typename T> unsigned mln::image3d< T >::nslis () const [inline]`

Give the number of slices.

Definition at line 527 of file core/image/image3d.hh.

10.213.4.14 `template<typename T> T & mln::image3d< T >::operator() (const point3d & p) [inline]`

Read-write access to the image value located at point p.

Definition at line 482 of file core/image/image3d.hh.

References mln::image3d< T >::has().

10.213.4.15 `template<typename T> const T & mln::image3d< T >::operator() (const point3d & p) const [inline]`

Read-only access to the image value located at point p.

Definition at line 473 of file core/image/image3d.hh.

References mln::image3d< T >::has().

10.213.4.16 `template<typename T> point3d mln::image3d< T >::point_at_index (unsigned o) const [inline]`

Give the point corresponding to the offset o.

Definition at line 583 of file core/image/image3d.hh.

References mln::image3d< T >::nelements().

10.213.4.17 `template<typename T> const box3d & mln::image3d< T >::vbbox () const [inline]`

virtual box, i.e., box including the virtual border

Definition at line 437 of file core/image/image3d.hh.

10.214 mln::interpolated< I, F > Struct Template Reference

Makes the underlying image being accessed with floating coordinates.

```
#include <interpolated.hh>
```

Inherits `image_identity< I, I::domain_t, interpolated< I, F > >`.

Public Types

- `typedef I::lvalue lvalue`
Return type of read-write access.
- `typedef I::psite psite`
Point_Site associated type.
- `typedef I::rvalue rvalue`
Return type of read-only access.
- `typedef interpolated< tag::image_< I >, F > skeleton`
Skeleton.
- `typedef I::value value`
Value associated type.

Public Member Functions

- `template<typename C >`
`bool has (const mln::algebra::vec< I::psite::dim, C > &v) const`
Test if a pixel value is accessible at v.
- `interpolated (I &ima)`
Constructors.
- `bool is_valid () const`
Test if this image has been initialized.

10.214.1 Detailed Description

`template<typename I, template< class > class F> struct mln::interpolated< I, F >`

Makes the underlying image being accessed with floating coordinates.

Definition at line 84 of file `interpolated.hh`.

10.214.2 Member Typedef Documentation

10.214.2.1 `template<typename I , template< class > class F> typedef I ::lvalue
mln::interpolated< I, F >::lvalue`

Return type of read-write access.

Definition at line 98 of file `interpolated.hh`.

10.214.2.2 `template<typename I , template< class > class F> typedef I ::psite mln::interpolated< I, F >::psite`

Point_Site associated type.

Definition at line 92 of file interpolated.hh.

10.214.2.3 `template<typename I , template< class > class F> typedef I ::rvalue mln::interpolated< I, F >::rvalue`

Return type of read-only access.

Definition at line 101 of file interpolated.hh.

10.214.2.4 `template<typename I , template< class > class F> typedef interpolated< tag::image_<I>, F > mln::interpolated< I, F >::skeleton`

Skeleton.

Definition at line 104 of file interpolated.hh.

10.214.2.5 `template<typename I , template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

[Value](#) associated type.

Definition at line 95 of file interpolated.hh.

10.214.3 Constructor & Destructor Documentation

10.214.3.1 `template<typename I , template< class > class F> mln::interpolated< I, F >::interpolated (I & ima) [inline]`

Constructors.

FIXME: don't we want a 'const' here?

Definition at line 156 of file interpolated.hh.

10.214.4 Member Function Documentation

10.214.4.1 `template<typename I , template< class > class F> template<typename C > bool mln::interpolated< I, F >::has (const mln::algebra::vec< I::psite::dim, C > & v) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 189 of file interpolated.hh.

10.214.4.2 `template<typename I , template< class > class F> bool mln::interpolated< I, F >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 180 of file interpolated.hh.

10.215 mln::io::dicom::dicom_header Struct Reference

Store dicom file header.

```
#include <get_header.hh>
```

10.215.1 Detailed Description

Store dicom file header.

Definition at line 59 of file dicom/get_header.hh.

10.216 mln::io::dump::dump_header Struct Reference

Store dump file header.

```
#include <get_header.hh>
```

10.216.1 Detailed Description

Store dump file header.

Definition at line 53 of file dump/get_header.hh.

10.217 mln::io::fld::fld_header Struct Reference

Define the header structure of an AVS field data file.

```
#include <header.hh>
```

10.217.1 Detailed Description

Define the header structure of an AVS field data file.

Definition at line 45 of file header.hh.

10.218 mln::io::raw::raw_header Struct Reference

Store raw file header.

```
#include <get_header.hh>
```

10.218.1 Detailed Description

Store raw file header.

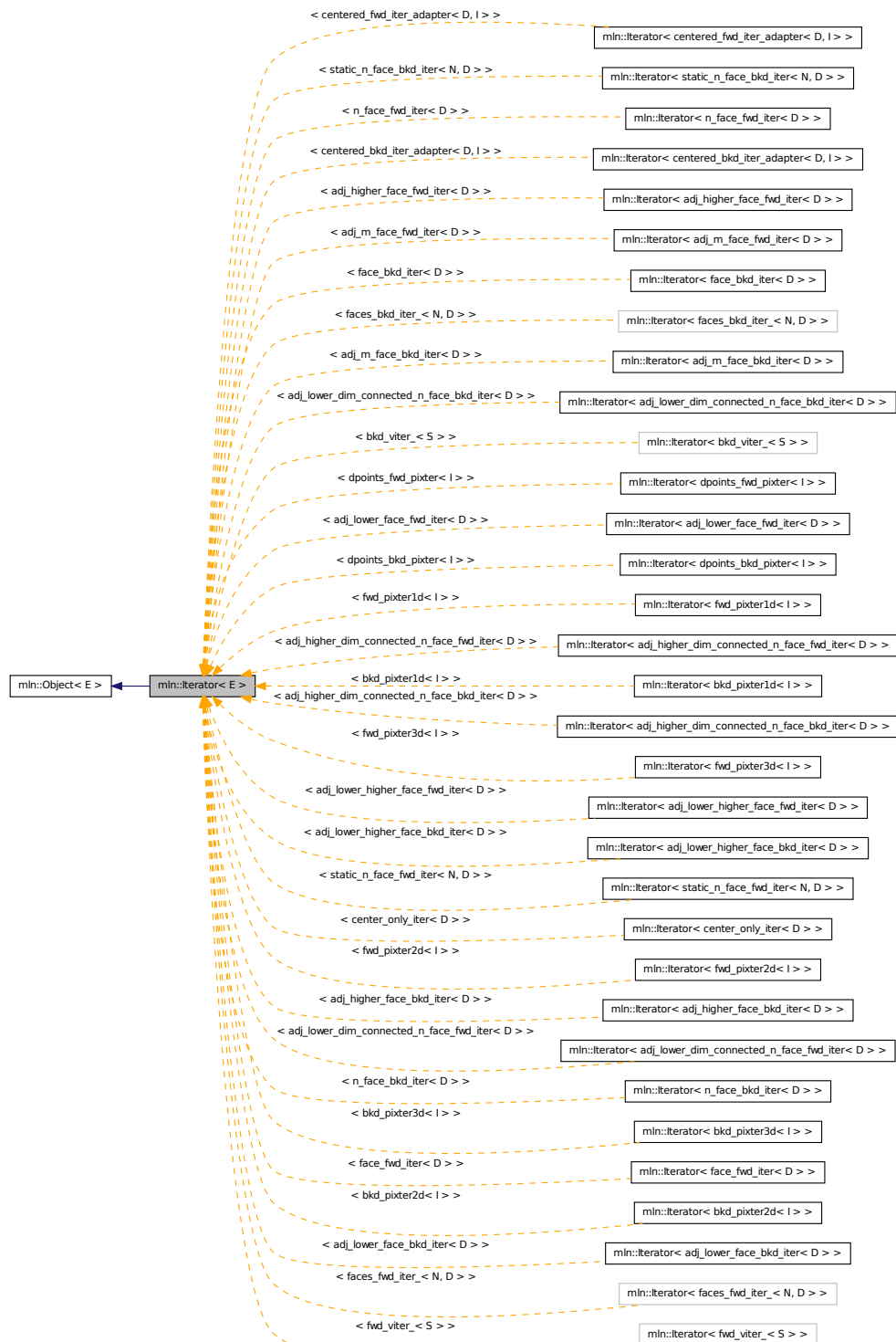
Definition at line 53 of file raw/get_header.hh.

10.219 mln::Iterator< E > Struct Template Reference

Base class for implementation classes that are iterators.

```
#include <iterator.hh>
```

Inheritance diagram for mln::Iterator< E >:



Public Member Functions

- void [next](#) ()

Go to the next element.

10.219.1 Detailed Description

`template<typename E> struct mln::Iterator< E >`

Base class for implementation classes that are iterators.

See also

[mln::doc::Iterator](#) for a complete documentation of this class contents.

Definition at line 75 of file iterator.hh.

10.219.2 Member Function Documentation

10.219.2.1 `template<typename E > void mln::Iterator< E >::next ()`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

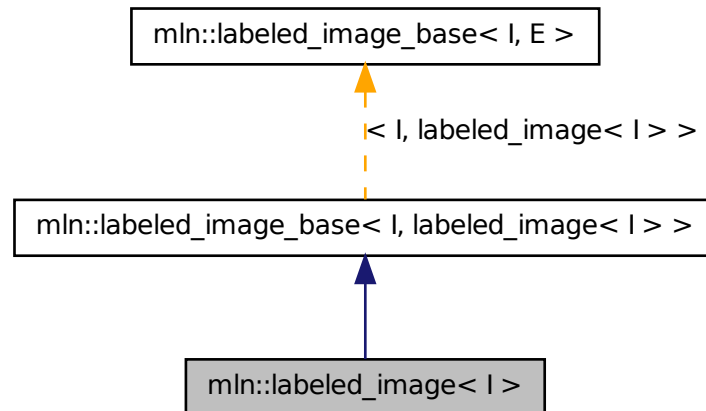
Definition at line 102 of file iterator.hh.

10.220 mln::labeled_image< I > Class Template Reference

Morpher providing an improved interface for labeled image.

```
#include <labeled_image.hh>
```

Inheritance diagram for `mln::labeled_image< I >`:



Public Types

- typedef `accu::shape::bbox< typename I::psite >::result` `bbox_t`
Type of the bounding component bounding boxes.
- typedef `labeled_image< tag::image_< I > >` `skeleton`
Skeleton.

Public Member Functions

- const `bbox_t` & `bbox` (const typename I::value & `label`) const
Return the bounding box of the component `label`.
- const `util::array< bbox_t >` & `bboxes` () const
Return the component bounding boxes.
- I::value `nlabels` () const
Return the number of labels;.
- `p_if< mln_box(I), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > > >` `subdomain` (const typename I::value & `label`) const
Return the domain of the component with label `label`.
- `labeled_image` ()

*Constructors**Constructor without argument.*

- [labeled_image](#) (const I &ima, const typename I::value &nlabels)
Constructor from an image `ima` and the number of labels `nlabels`.
- [labeled_image](#) (const I &ima, const typename I::value &nlabels, const [util::array](#)< mln_box(I)> &bboxes)
Constructor from an image `ima`, the number of labels `nlabels` and the object bounding boxes.
- void [relabel](#) (const [Function_v2v](#)< F > &f)
Relabel according to a function.
- void [relabel](#) (const [Function_v2b](#)< F > &f)
Labels may be removed.

Protected Member Functions

- void [update_data](#) (const fun::i2v::array< typename I::value > &relabel_fun)
Update bounding boxes information.

10.220.1 Detailed Description

```
template<typename I> class mln::labeled_image< I >
```

Morpher providing an improved interface for labeled image.

Template Parameters

I The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 105 of file labeled_image.hh.

10.220.2 Member Typedef Documentation

10.220.2.1 `typedef accu::shape::bbox<typename I ::psite>::result mln::labeled_image_base< I, labeled_image< I > >::bbox_t [inherited]`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.220.2.2 `template<typename I> typedef labeled_image< tag::image_<I> > mln::labeled_image< I >::skeleton`

Skeleton.

Definition at line 113 of file labeled_image.hh.

10.220.3 Constructor & Destructor Documentation

10.220.3.1 `template<typename I> mln::labeled_image<I>::labeled_image () [inline]`

Constructors

Constructor without argument.

Definition at line 193 of file labeled_image.hh.

10.220.3.2 `template<typename I> mln::labeled_image<I>::labeled_image (const I & ima, const typename I::value & nlabels) [inline]`

Constructor from an image `ima` and the number of labels `nlabels`.

Definition at line 199 of file labeled_image.hh.

10.220.3.3 `template<typename I> mln::labeled_image<I>::labeled_image (const I & ima, const typename I::value & nlabels, const util::array<mln_box(I)> & bboxes) [inline]`

Constructor from an image `ima`, the number of labels `nlabels` and the object bounding boxes.

Definition at line 206 of file labeled_image.hh.

References `mln::data::compute()`.

10.220.4 Member Function Documentation

10.220.4.1 `const bbox_t& mln::labeled_image_base<I, labeled_image<I>>::bbox (const typename I::value & label) const [inherited]`

Return the bounding box of the component `label`.

10.220.4.2 `const util::array<bbox_t>& mln::labeled_image_base<I, labeled_image<I>>::bboxes () const [inherited]`

Return the component bounding boxes.

10.220.4.3 `I::value mln::labeled_image_base<I, labeled_image<I>>::nlabels () const [inherited]`

Return the number of labels;.

10.220.4.4 `void mln::labeled_image_base<I, labeled_image<I>>::relabel (const Function_v2b<F> & f) [inherited]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

10.220.4.5 void mln::labeled_image_base< I, labeled_image< I > >::relabel (const Function_v2v< F > & f) [inherited]

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

10.220.4.6 p_if<mln_box(I), fun::eq_v2b_expr_<pw::value_<I>, pw::cst_<typename I::value> > > mln::labeled_image_base< I, labeled_image< I > >::subdomain (const typename I::value & label) const [inherited]

Return the domain of the component with label `label`.

10.220.4.7 void mln::labeled_image_base< I, labeled_image< I > >::update_data (const fun::i2v::array< typename I::value > & relabel_fun) [protected, inherited]

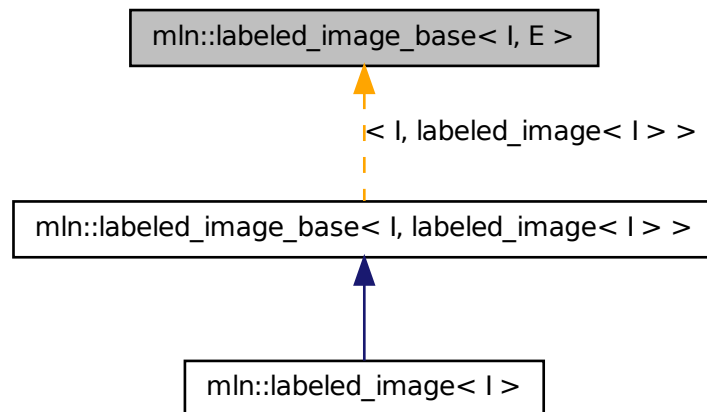
Update bounding boxes information.

10.221 mln::labeled_image_base< I, E > Class Template Reference

Base class Morpher providing an improved interface for labeled image.

```
#include <labeled_image_base.hh>
```

Inheritance diagram for mln::labeled_image_base< I, E >:



Public Types

- typedef [accu::shape::bbox](#)< typename I::psite >::result [bbox_t](#)
Type of the bounding component bounding boxes.

Public Member Functions

- const [bbox_t](#) & [bbox](#) (const typename I::value &[label](#)) const
Return the bounding box of the component [label](#).
- const [util::array](#)< [bbox_t](#) > & [bboxes](#) () const
Return the component bounding boxes.
- I::value [nlabels](#) () const
Return the number of labels;.
- [p_if](#)< mln_box(I), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > > >
[subdomain](#) (const typename I::value &[label](#)) const
Return the domain of the component with label [label](#).
- [labeled_image_base](#) ()
Constructors
Constructor without argument.
- template<typename F >
void [relabel](#) (const [Function_v2v](#)< F > &f)
Relabel according to a function.
- template<typename F >
void [relabel](#) (const [Function_v2b](#)< F > &f)
Labels may be removed.

Protected Member Functions

- void [update_data](#) (const fun::i2v::array< typename I::value > &[relabel_fun](#))
Update bounding boxes information.

10.221.1 Detailed Description

template<typename I, typename E> class mln::labeled_image_base< I, E >

Base class Morpher providing an improved interface for labeled image.

Template Parameters

I The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 116 of file labeled_image_base.hh.

10.221.2 Member Typedef Documentation

10.221.2.1 `template<typename I, typename E> typedef accu::shape::bbox<typename I::psite>::result mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.221.3 Constructor & Destructor Documentation

10.221.3.1 `template<typename I, typename E > mln::labeled_image_base< I, E >::labeled_image_base () [inline]`

Constructors

Constructor without argument.

Definition at line 217 of file labeled_image_base.hh.

10.221.4 Member Function Documentation

10.221.4.1 `template<typename I, typename E > const labeled_image_base< I, E >::bbox_t & mln::labeled_image_base< I, E >::bbox (const typename I::value & label) const`

Return the bounding box of the component `label`.

Definition at line 305 of file labeled_image_base.hh.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

10.221.4.2 `template<typename I, typename E > const util::array< typename labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes () const`

Return the component bounding boxes.

Definition at line 313 of file labeled_image_base.hh.

10.221.4.3 `template<typename I, typename E > I::value mln::labeled_image_base< I, E >::nlabels () const [inline]`

Return the number of labels;.

Definition at line 273 of file labeled_image_base.hh.

10.221.4.4 `template<typename I , typename E > template<typename F > void
mln::labeled_image_base< I, E >::relabel (const Function_v2b< F > & f)
[inline]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

Definition at line 252 of file labeled_image_base.hh.

References mln::labeling::relabel_inplace(), mln::make::relabelfun(), and mln::labeled_image_base< I, E >::update_data().

10.221.4.5 `template<typename I , typename E > template<typename F > void
mln::labeled_image_base< I, E >::relabel (const Function_v2v< F > & f)
[inline]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

Definition at line 226 of file labeled_image_base.hh.

References mln::labeling::relabel_inplace(), mln::make::relabelfun(), and mln::labeled_image_base< I, E >::update_data().

10.221.4.6 `template<typename I, typename E > p_if< mln_box(I), fun::eq_v2b_expr_<
pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, E
>::subdomain (const typename I::value & label) const`

Return the domain of the component with label `label`.

Definition at line 322 of file labeled_image_base.hh.

References mln::labeled_image_base< I, E >::bbox().

10.221.4.7 `template<typename I, typename E > void mln::labeled_image_base< I, E
>::update_data (const fun::i2v::array< typename I::value > & relabel_fun)
[protected]`

Update bounding boxes information.

Definition at line 281 of file labeled_image_base.hh.

References mln::util::array< T >::size().

Referenced by mln::labeled_image_base< I, E >::relabel().

10.222 mln::lazy_image< I, F, B > Struct Template Reference

Image values are computed on the fly.

```
#include <lazy_image.hh>
```

Inherits image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B > >.

Public Types

- typedef F::result [lvalue](#)
Return type of read-write access.
- typedef F::result [rvalue](#)
Return type of read access.
- typedef [lazy_image](#)< tag::image_< I >, F, B > [skeleton](#)
Skeleton.

Public Member Functions

- const [box](#)< typename I::psite > & [domain](#) () const
Return domain of lazyd_image.
- bool [has](#) (const typename I::psite &) const
Test if a pixel value is accessible at p.
- [lazy_image](#) (const F &fun, const B &[box](#))
Constructors.
- [lazy_image](#) ()
Constructors.
- F::result [operator\(\)](#) (const typename F::input &x) const
Read-only access of pixel value at F::input x.
- [lvalue operator\(\)](#) (const typename I::psite &p)
Read and "write if possible" access of pixel value at point site p.
- F::result [operator\(\)](#) (const typename F::input &x)
Read and "write if possible" access of pixel value at F::input x.
- [rvalue operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.

10.222.1 Detailed Description

template<typename I, typename F, typename B> struct mln::lazy_image< I, F, B >

[Image](#) values are computed on the fly. The parameter I is the type of image. The parameter F is the type of function. The parameter B is the type of box.

This image class take a functor fun and a box box. Access to ima(p) where p include box return fun(b) lazily.

Definition at line 92 of file lazy_image.hh.

10.222.2 Member Typedef Documentation

10.222.2.1 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue`

Return type of read-write access.

Definition at line 104 of file lazy_image.hh.

10.222.2.2 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue`

Return type of read access.

Definition at line 101 of file lazy_image.hh.

10.222.2.3 `template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton`

Skeleton.

Definition at line 107 of file lazy_image.hh.

10.222.3 Constructor & Destructor Documentation

10.222.3.1 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ()`

Constructors.

10.222.3.2 `template<typename I , typename F, typename B> mln::lazy_image< I, F, B >::lazy_image (const F & fun, const B & box) [inline]`

Constructors.

Definition at line 161 of file lazy_image.hh.

10.222.4 Member Function Documentation

10.222.4.1 `template<typename I , typename F , typename B > const box< typename I::psite > & mln::lazy_image< I, F, B >::domain () const [inline]`

Return domain of lazyd_image.

Definition at line 226 of file lazy_image.hh.

10.222.4.2 `template<typename I, typename F , typename B > bool mln::lazy_image< I, F, B >::has (const typename I::psite & p) const [inline]`

Test if a pixel value is accessible at p.

Definition at line 175 of file lazy_image.hh.

10.222.4.3 `template<typename I, typename F, typename B > lazy_image< I, F, B >::rvalue
mln::lazy_image< I, F, B >::operator() (const typename I::psite & p) const
[inline]`

Read-only access of pixel value at point site p.

Definition at line 210 of file lazy_image.hh.

10.222.4.4 `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B
>::operator() (const typename F::input & x) [inline]`

Read and "write if possible" access of pixel value at F::input x.

Definition at line 197 of file lazy_image.hh.

10.222.4.5 `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B
>::operator() (const typename F::input & x) const [inline]`

Read-only access of pixel value at F::input x.

Definition at line 183 of file lazy_image.hh.

10.222.4.6 `template<typename I, typename F, typename B > lazy_image< I, F, B >::lvalue
mln::lazy_image< I, F, B >::operator() (const typename I::psite & p) [inline]`

Read and "write if possible" access of pixel value at point site p.

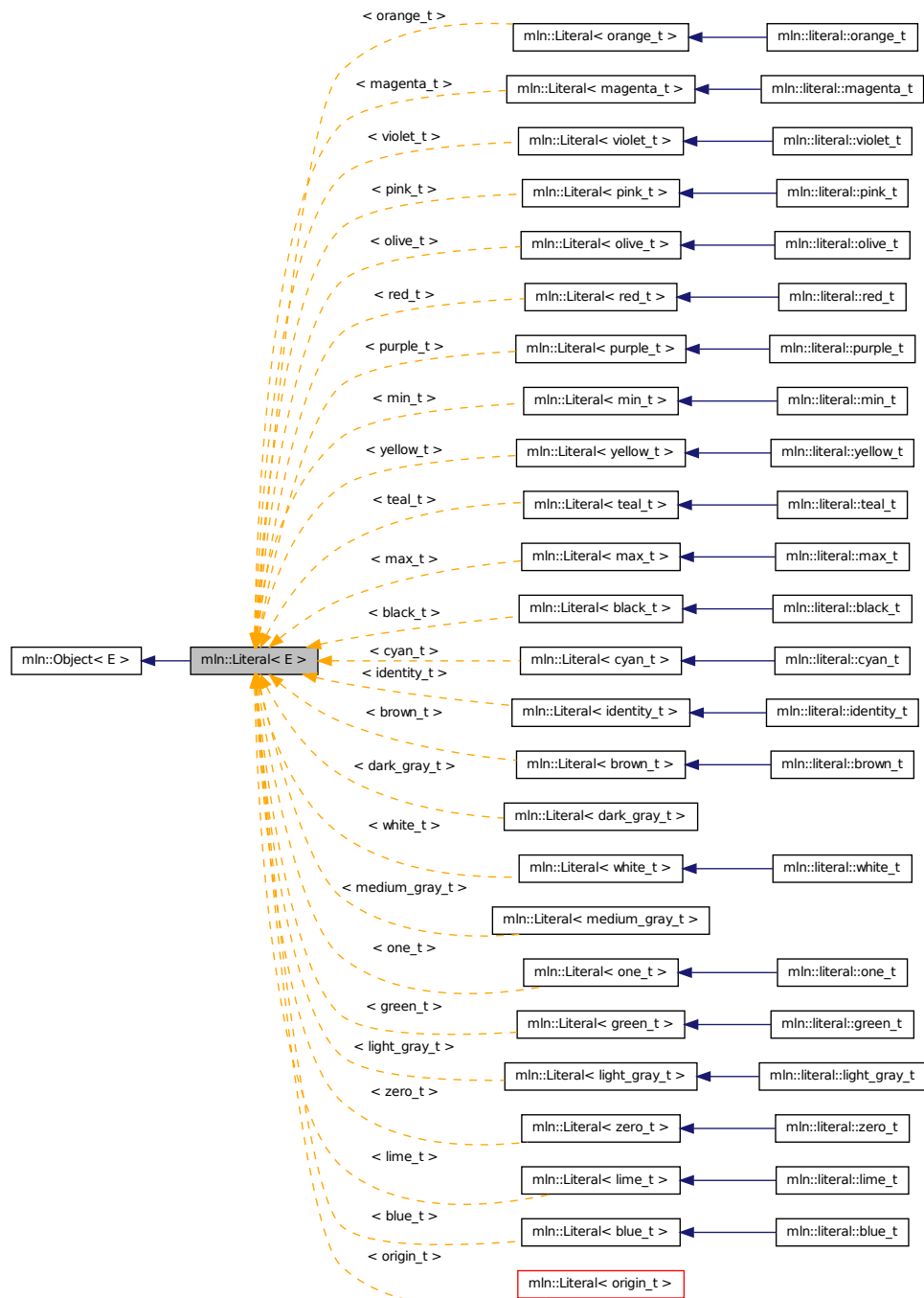
Definition at line 218 of file lazy_image.hh.

10.223 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

`#include <literal.hh>`

Inheritance diagram for mln::Literal< E >:



10.223.1 Detailed Description

```
template<typename E> struct mln::Literal< E >
```

Base class for implementation classes of literals.

See also

mln::doc::Literal for a complete documentation of this class contents.

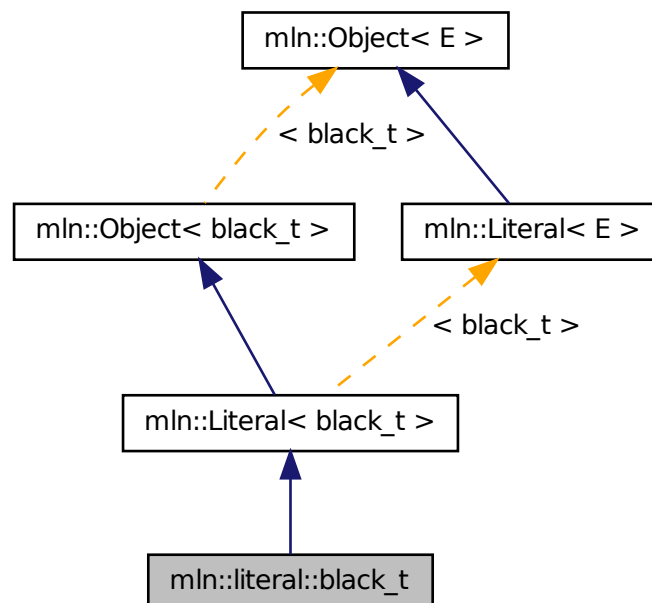
Definition at line 56 of file literal.hh.

10.224 mln::literal::black_t Struct Reference

Type of literal black.

```
#include <black.hh>
```

Inheritance diagram for mln::literal::black_t:



10.224.1 Detailed Description

Type of literal black.

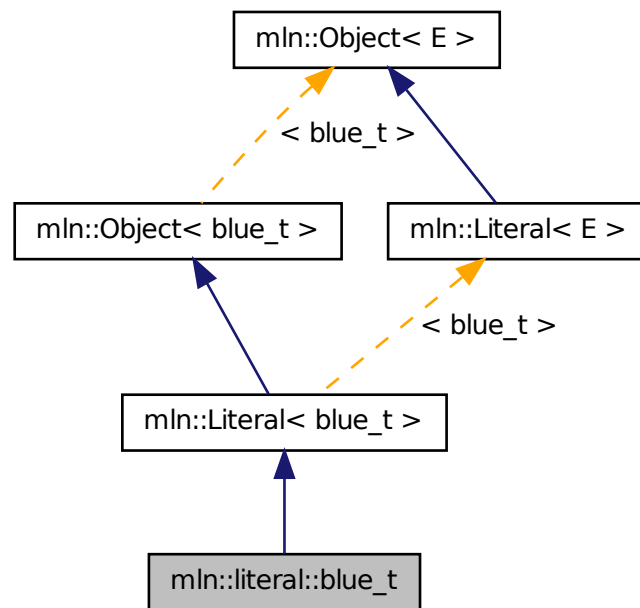
Definition at line 43 of file black.hh.

10.225 mln::literal::blue_t Struct Reference

Type of literal blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue_t:



10.225.1 Detailed Description

Type of literal blue.

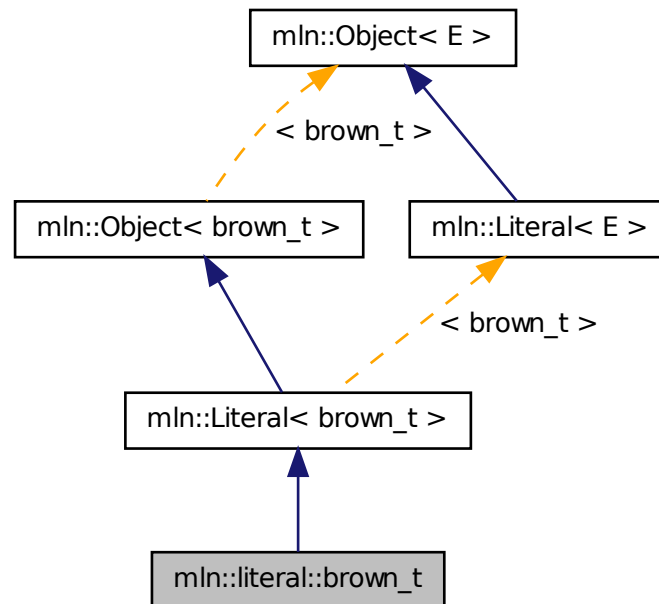
Definition at line 53 of file `colors.hh`.

10.226 mln::literal::brown_t Struct Reference

Type of literal brown.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::brown_t:



10.226.1 Detailed Description

Type of literal brown.

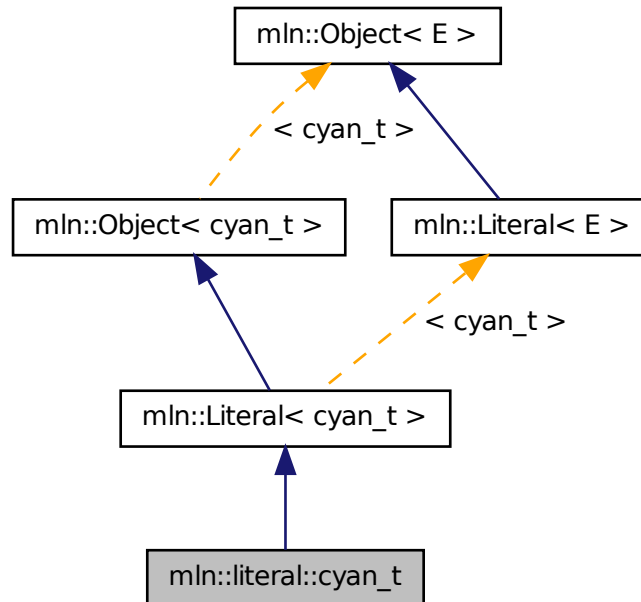
Definition at line 58 of file `colors.hh`.

10.227 mln::literal::cyan_t Struct Reference

Type of literal cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan_t:



10.227.1 Detailed Description

Type of literal cyan.

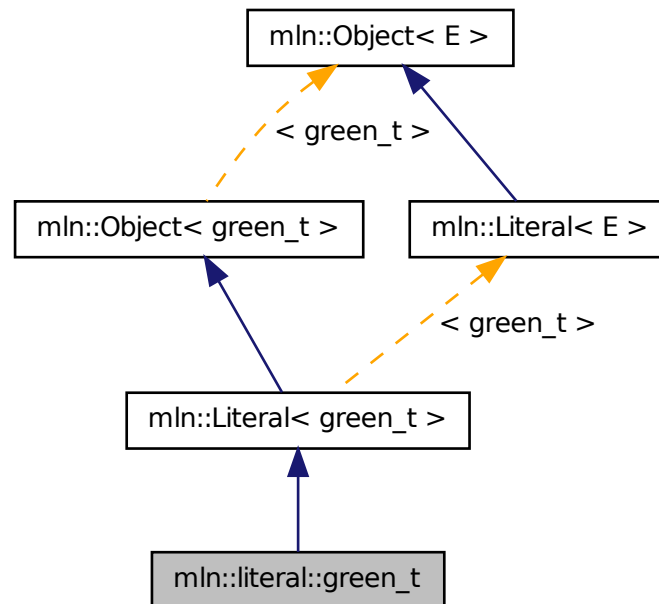
Definition at line 93 of file colors.hh.

10.228 mln::literal::green_t Struct Reference

Type of literal green.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::green_t:



10.228.1 Detailed Description

Type of literal green.

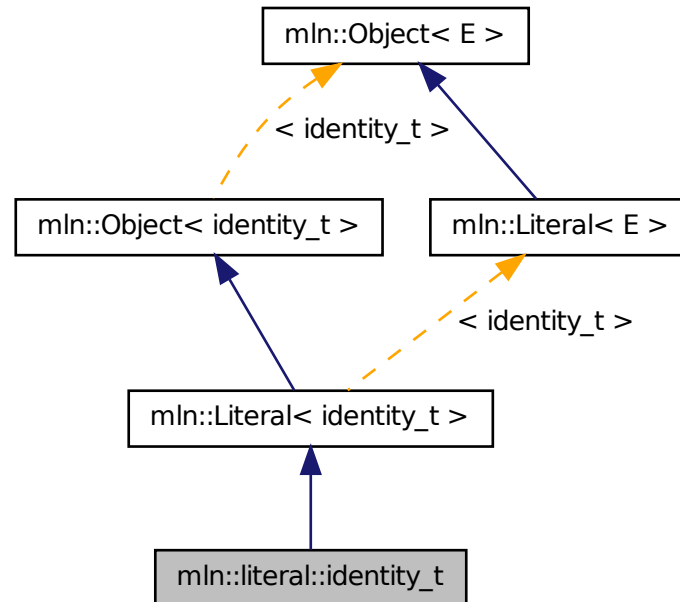
Definition at line 48 of file colors.hh.

10.229 mln::literal::identity_t Struct Reference

Type of literal identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity_t:



10.229.1 Detailed Description

Type of literal identity.

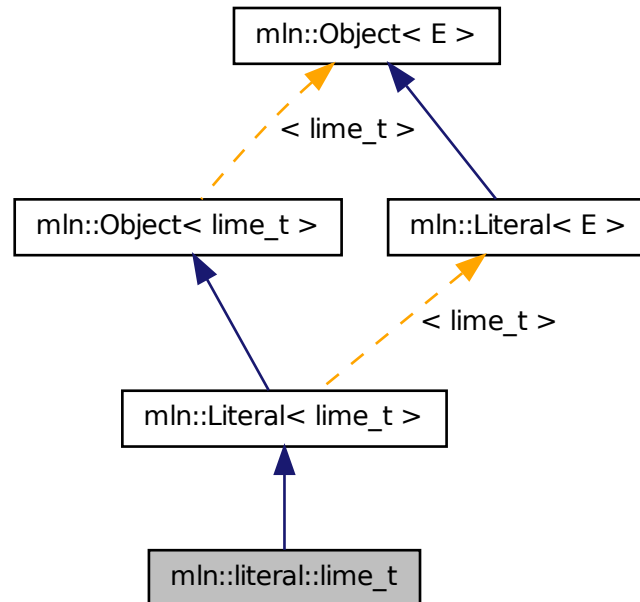
Definition at line 43 of file identity.hh.

10.230 mln::literal::light_gray_t Struct Reference

Type of literal grays.

```
#include <grays.hh>
```


Inheritance diagram for mln::literal::lime_t:



10.231.1 Detailed Description

Type of literal lime.

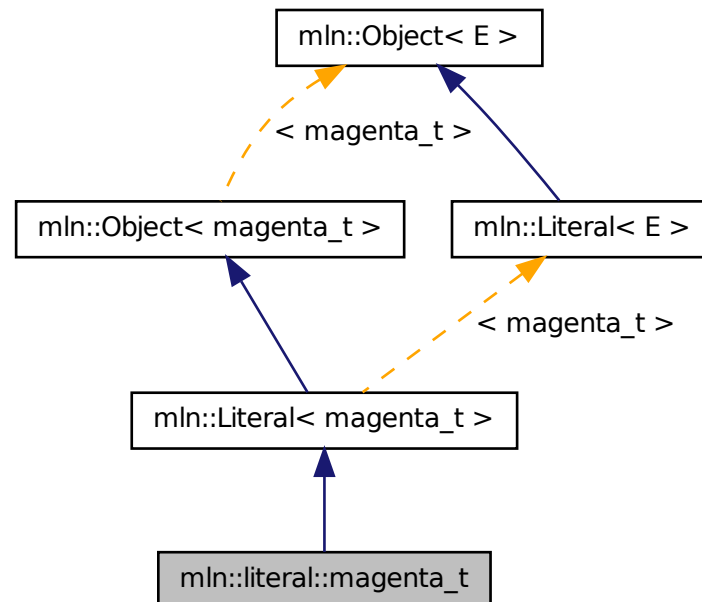
Definition at line 63 of file colors.hh.

10.232 mln::literal::magenta_t Struct Reference

Type of literal magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta_t:



10.232.1 Detailed Description

Type of literal magenta.

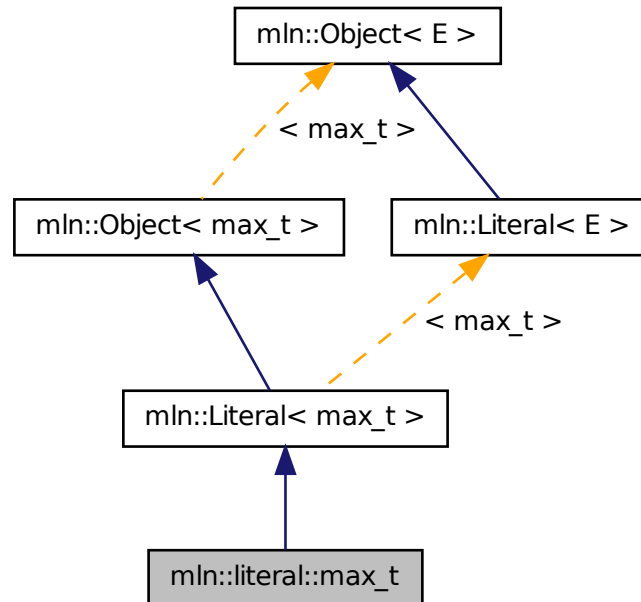
Definition at line 98 of file colors.hh.

10.233 mln::literal::max_t Struct Reference

Type of literal max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max_t:



10.233.1 Detailed Description

Type of literal max.

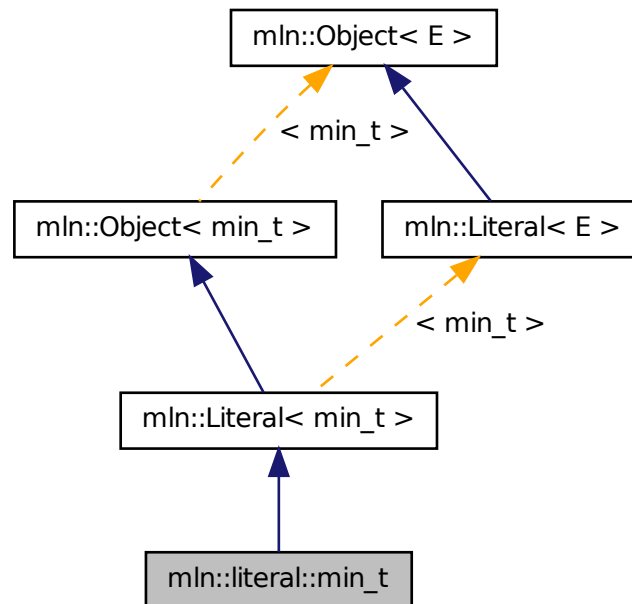
Definition at line 44 of file literal/max.hh.

10.234 mln::literal::min_t Struct Reference

Type of literal min.

```
#include <min.hh>
```

Inheritance diagram for mln::literal::min_t:



10.234.1 Detailed Description

Type of literal min.

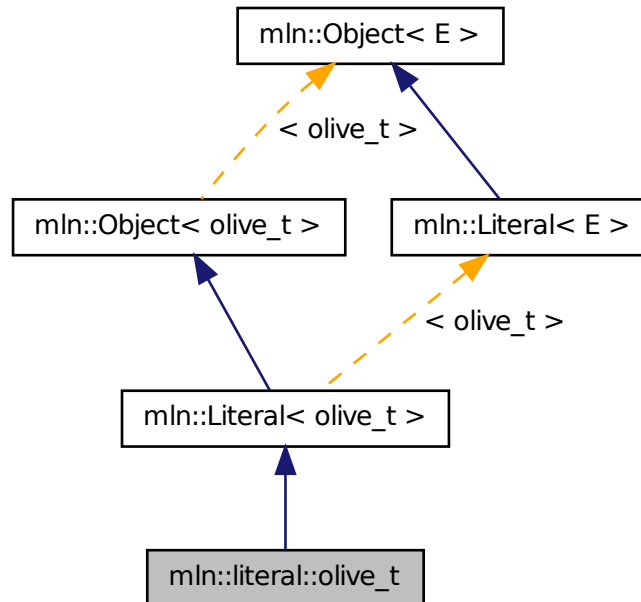
Definition at line 44 of file literal/min.hh.

10.235 mln::literal::olive_t Struct Reference

Type of literal olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive_t:



10.235.1 Detailed Description

Type of literal olive.

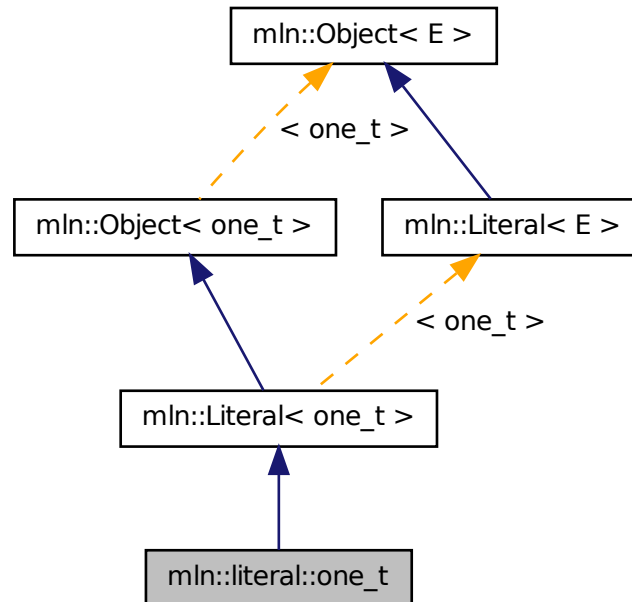
Definition at line 108 of file colors.hh.

10.236 mln::literal::one_t Struct Reference

Type of literal one.

```
#include <one.hh>
```

Inheritance diagram for mln::literal::one_t:



10.236.1 Detailed Description

Type of literal one.

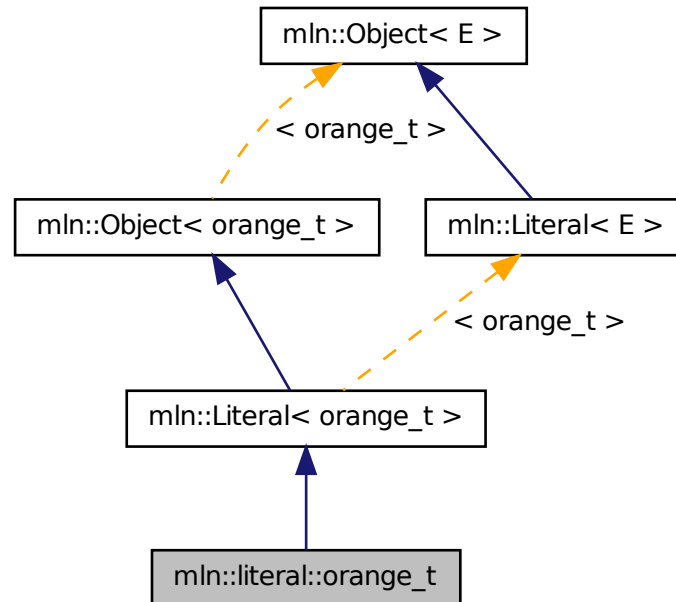
Definition at line 44 of file one.hh.

10.237 mln::literal::orange_t Struct Reference

Type of literal orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange_t:



10.237.1 Detailed Description

Type of literal orange.

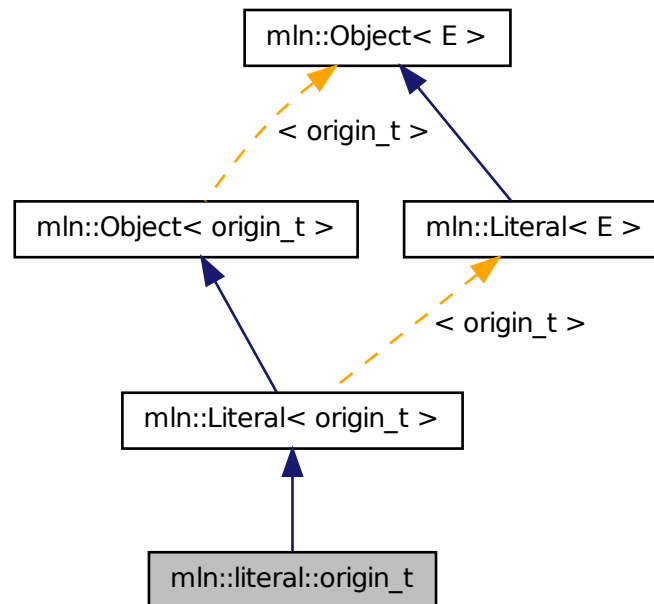
Definition at line 68 of file colors.hh.

10.238 mln::literal::origin_t Struct Reference

Type of literal origin.

```
#include <origin.hh>
```


Inheritance diagram for mln::literal::origin_t:



10.238.1 Detailed Description

Type of literal origin.

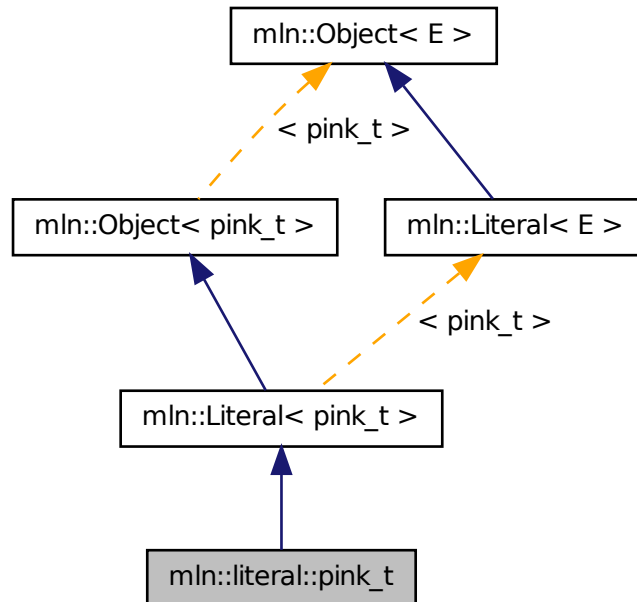
Definition at line 44 of file `origin.hh`.

10.239 mln::literal::pink_t Struct Reference

Type of literal pink.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::pink_t:



10.239.1 Detailed Description

Type of literal pink.

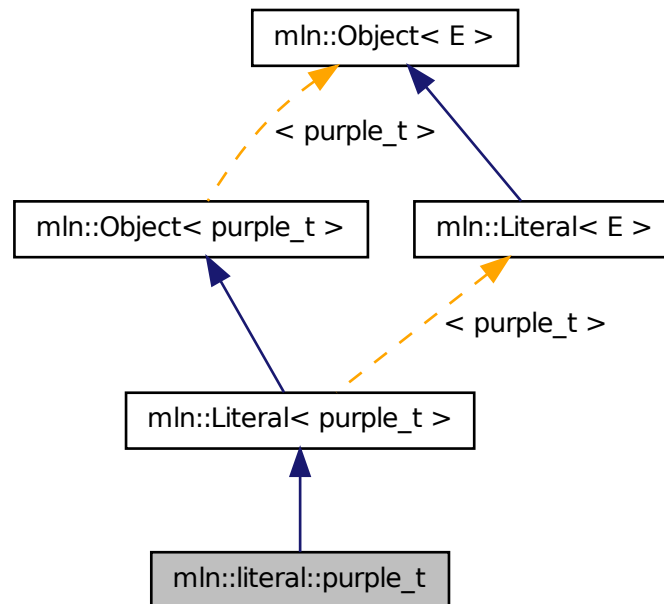
Definition at line 73 of file colors.hh.

10.240 mln::literal::purple_t Struct Reference

Type of literal purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple_t:



10.240.1 Detailed Description

Type of literal purple.

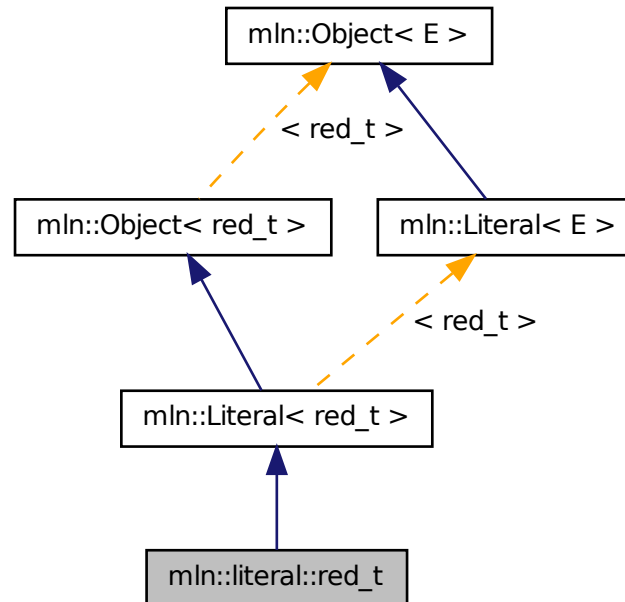
Definition at line 78 of file `colors.hh`.

10.241 mln::literal::red_t Struct Reference

Type of literal red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red_t:



10.241.1 Detailed Description

Type of literal red.

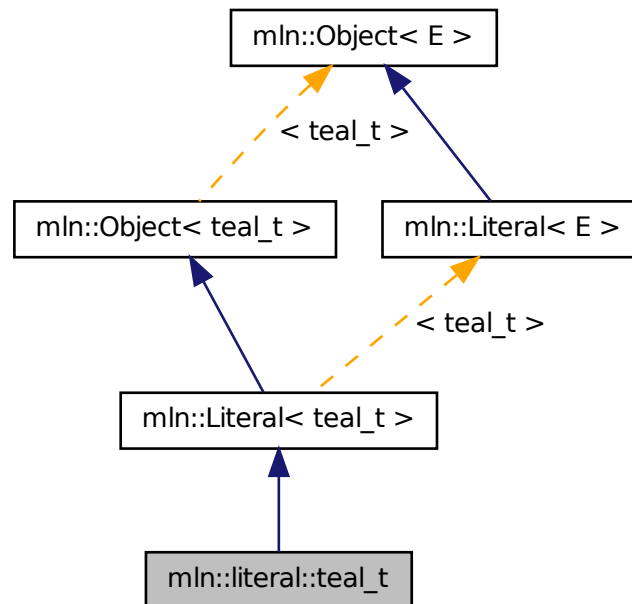
Definition at line 43 of file colors.hh.

10.242 mln::literal::teal_t Struct Reference

Type of literal teal.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::teal_t:



10.242.1 Detailed Description

Type of literal teal.

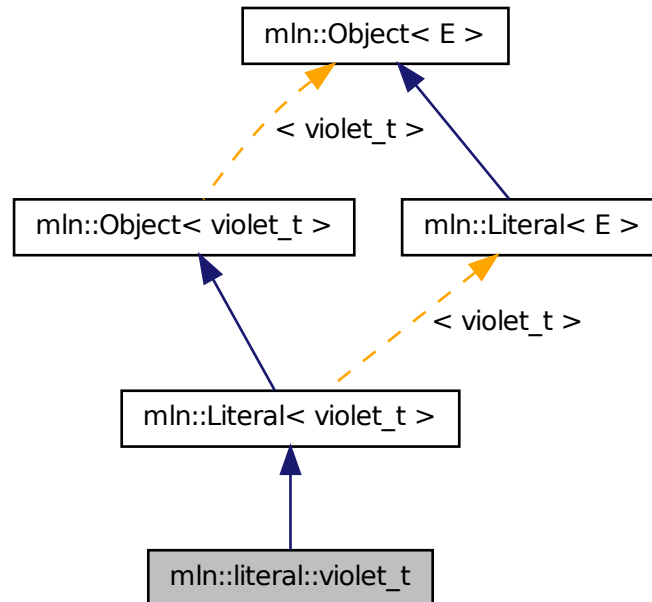
Definition at line 83 of file colors.hh.

10.243 mln::literal::violet_t Struct Reference

Type of literal violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet_t:



10.243.1 Detailed Description

Type of literal violet.

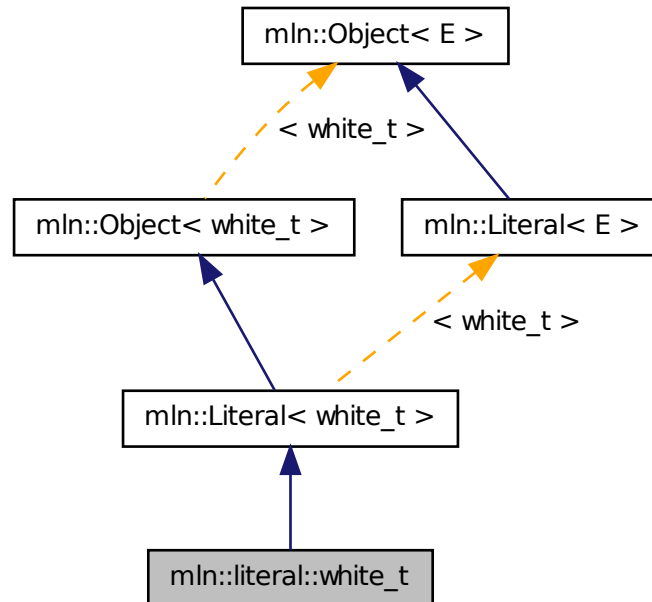
Definition at line 88 of file colors.hh.

10.244 mln::literal::white_t Struct Reference

Type of literal white.

```
#include <white.hh>
```

Inheritance diagram for mln::literal::white_t:



10.244.1 Detailed Description

Type of literal white.

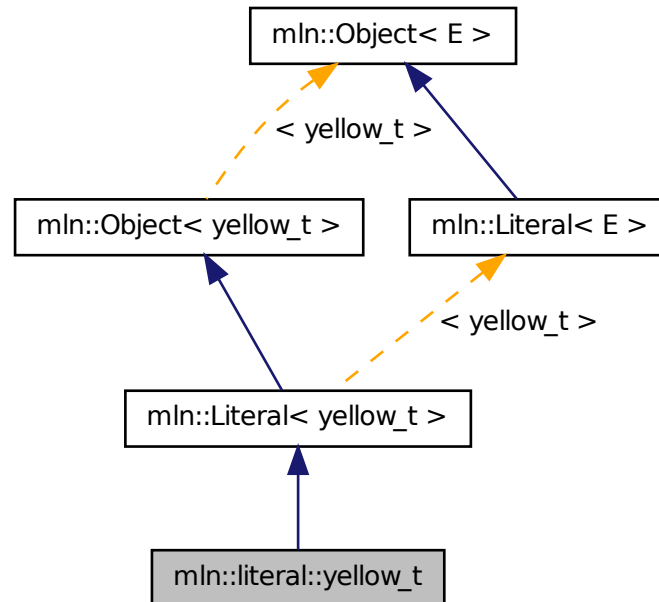
Definition at line 43 of file `white.hh`.

10.245 mln::literal::yellow_t Struct Reference

Type of literal yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow_t:



10.245.1 Detailed Description

Type of literal yellow.

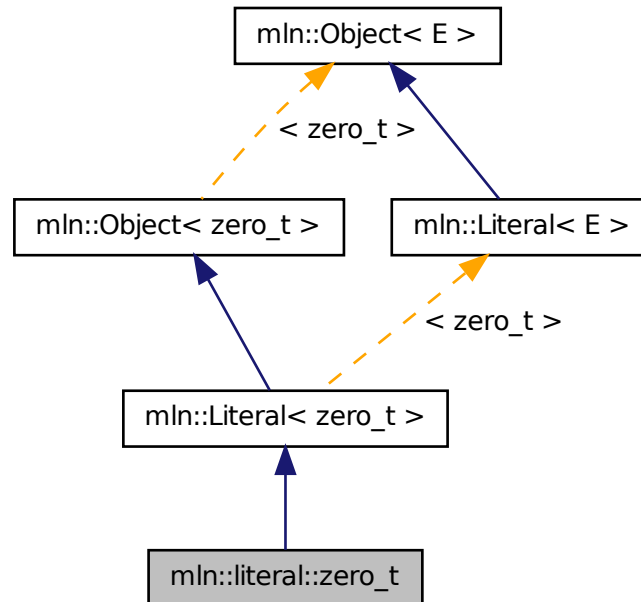
Definition at line 103 of file colors.hh.

10.246 mln::literal::zero_t Struct Reference

Type of literal zero.

```
#include <zero.hh>
```


Inheritance diagram for mln::literal::zero_t:



10.246.1 Detailed Description

Type of literal zero.

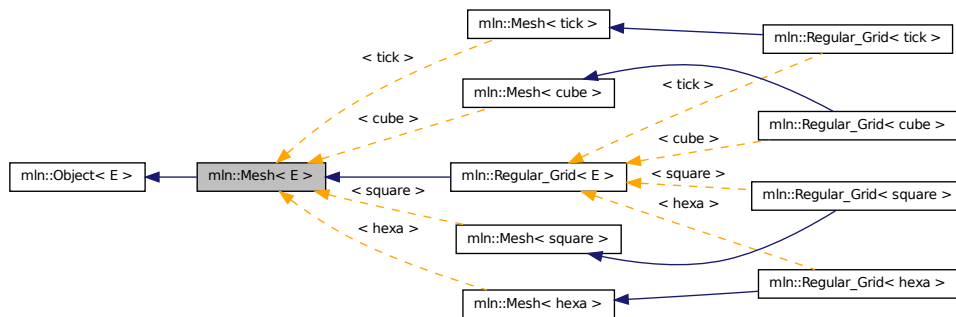
Definition at line 46 of file zero.hh.

10.247 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for `mln::Mesh< E >`:



10.247.1 Detailed Description

template<typename E> struct mln::Mesh< E >

Base class for implementation classes of meshes.

See also

`mln::doc::Mesh` for a complete documentation of this class contents.

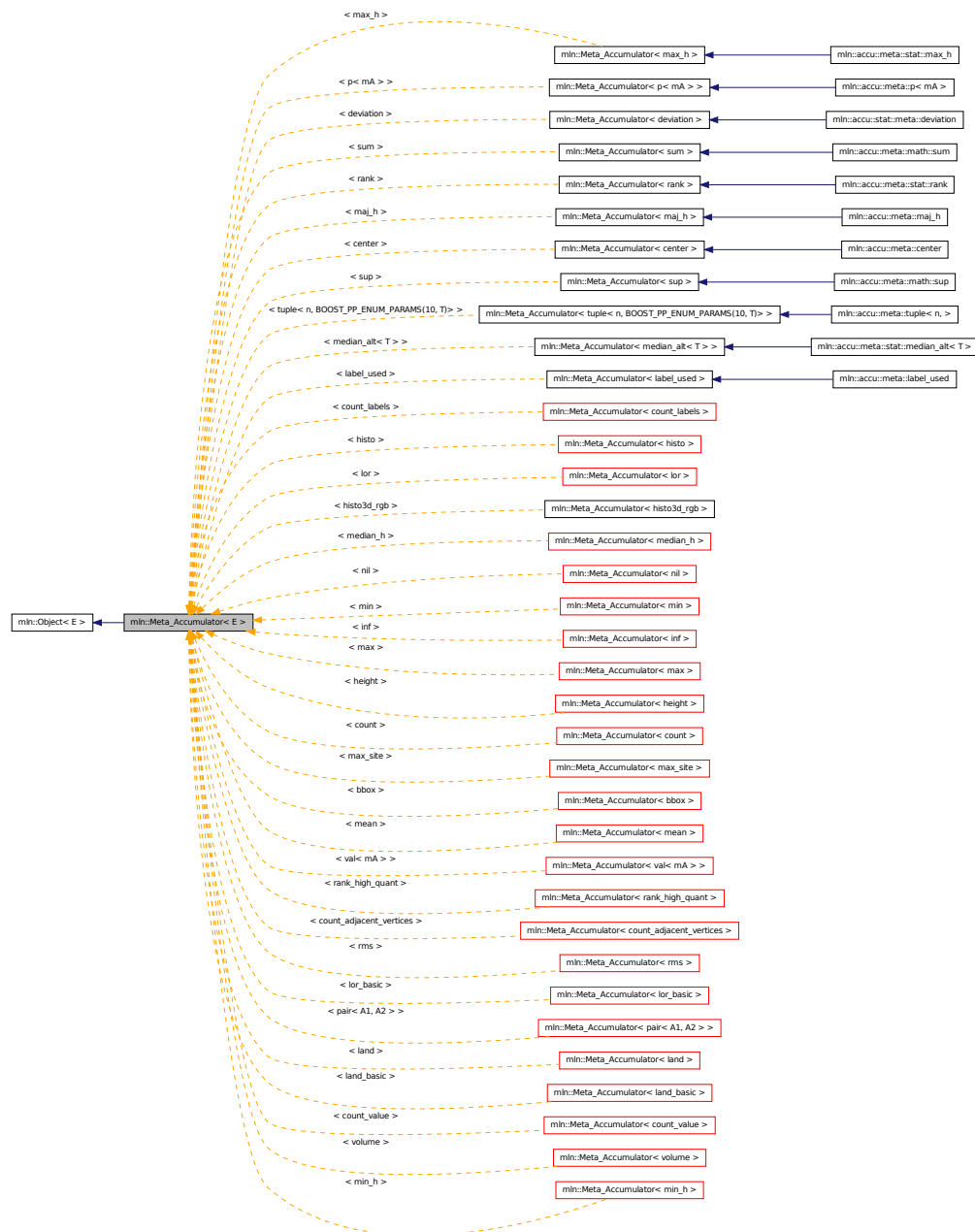
Definition at line 45 of file `mesh.hh`.

10.248 mln::Meta_Accumulator< E > Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inheritance diagram for mln::Meta_Accumulator< E >:



10.248.1 Detailed Description

`template<typename E> struct mln::Meta_Accumulator< E >`

Base class for implementation of meta accumulators. The parameter *E* is the exact type.

mln::doc::Meta_Accumulator for a complete documentation of this class contents.

10.249 mln::Meta_Function< E > Struct Template Reference

```
#include <meta_function.hh>
```

```
template<typename E> struct mln::Meta_Function< E >
```

mln::doc::Meta_Function for a complete documentation of this class contents.

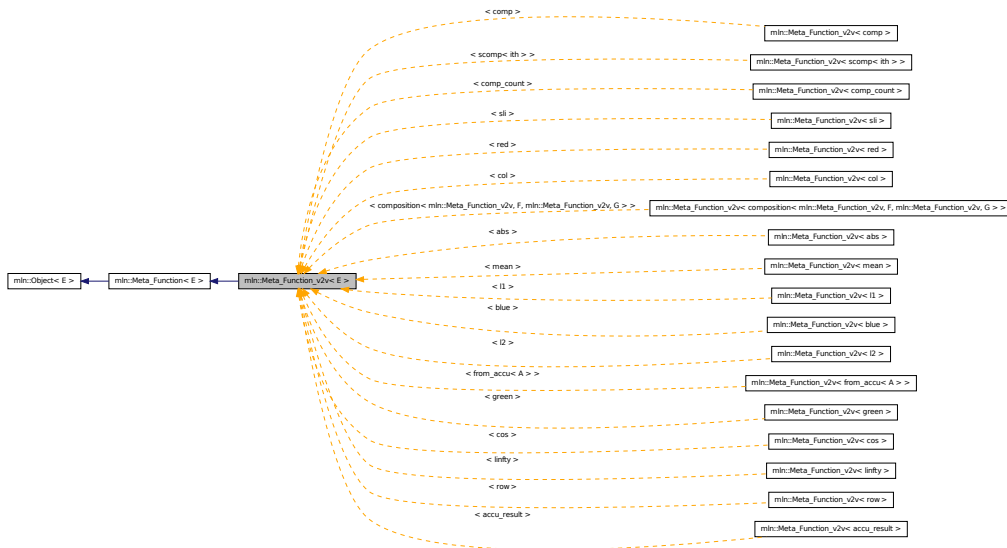
Generated on Thu Sep 15 2011 15:47:20 for Milena (Olena) by Doxygen

10.250 mln::Meta_Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta_Function_v2v< E >:



10.250.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_v2v< E >
```

Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.

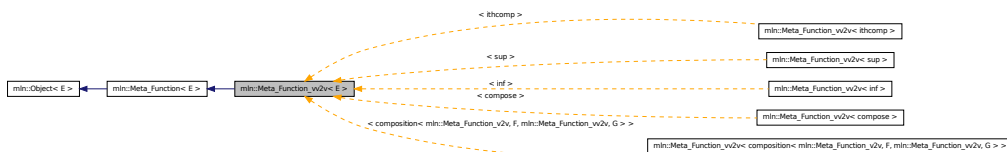
Definition at line 98 of file meta_function.hh.

10.251 mln::Meta_Function_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta_Function_vv2v< E >:



10.251.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_vv2v< E >
```

Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.
Definition at line 120 of file meta_function.hh.

10.252 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

10.252.1 Detailed Description

```
template<typename E1, typename E2, typename E3, typename E4 = true_, typename E5 = true_,  
typename E6 = true_, typename E7 = true_, typename E8 = true_> struct mln::metal::ands< E1,  
E2, E3, E4, E5, E6, E7, E8 >
```

Ands type.

Definition at line 51 of file ands.hh.

10.253 mln::metal::converts_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherited by mln::metal::converts_to< T *, U * >.

10.253.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::converts_to< T, U >
```

"converts-to" check.

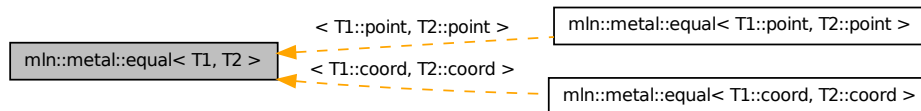
Definition at line 78 of file converts_to.hh.

10.254 mln::metal::equal< T1, T2 > Struct Template Reference

Definition of a static 'equal' test.

```
#include <equal.hh>
```

Inheritance diagram for mln::metal::equal< T1, T2 >:



10.254.1 Detailed Description

```
template<typename T1, typename T2> struct mln::metal::equal< T1, T2 >
```

Definition of a static 'equal' test. Check whether type T1 is exactly type T2.

Definition at line 49 of file equal.hh.

10.255 mln::metal::goes_to< T, U > Struct Template Reference

"goes-to" check.

```
#include <goes_to.hh>
```

10.255.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::goes_to< T, U >
```

"goes-to" check. FIXME: Doc!

Definition at line 53 of file goes_to.hh.

10.256 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

10.256.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is< T, U >
```

"is" check. Check whether T inherits from U.

Definition at line 64 of file is.hh.

10.257 mln::metal::is_a< T, M > Struct Template Reference

"is_a" check.

```
#include <is_a.hh>
```

10.257.1 Detailed Description

```
template<typename T, template< class > class M> struct mln::metal::is_a< T, M >
```

"is_a" check. Check whether T inherits from _CONCEPT_ M.

Definition at line 95 of file is_a.hh.

10.258 mln::metal::is_not< T, U > Struct Template Reference

"is_not" check.

```
#include <is_not.hh>
```

10.258.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is_not< T, U >
```

"is_not" check. FIXME: Doc!

Definition at line 52 of file is_not.hh.

10.259 mln::metal::is_not_a< T, M > Struct Template Reference

"is_not_a" static Boolean expression.

```
#include <is_not_a.hh>
```

10.259.1 Detailed Description

```
template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >
```

"is_not_a" static Boolean expression.

Definition at line 48 of file is_not_a.hh.

10.260 mln::morpho::attribute::card< I > Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits base< unsigned, card< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- unsigned [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.260.1 Detailed Description

`template<typename I> class mln::morpho::attribute::card< I >`

Cardinality accumulator class.

Definition at line 80 of file morpho/attribute/card.hh.

10.260.2 Member Function Documentation

10.260.2.1 `template<typename I> void mln::morpho::attribute::card< I >::init ()`
[[inline](#)]

Manipulators.

Definition at line 128 of file morpho/attribute/card.hh.

10.260.2.2 `template<typename I> bool mln::morpho::attribute::card< I >::is_valid () const`
[[inline](#)]

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file morpho/attribute/card.hh.

10.260.2.3 `void mln::Accumulator< card< I > >::take_as_init (const T & t)` [[inherited](#)]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.260.2.4 `void mln::Accumulator< card< I > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.260.2.5 `template<typename I > unsigned mln::morpho::attribute::card< I >::to_result ()`
const [inline]

Get the value of the accumulator.

Definition at line 189 of file `morpho/attribute/card.hh`.

10.261 mln::morpho::attribute::count_adjacent_vertices< I > Struct Template Reference

Count_Adjacent_Vertices accumulator class.

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count_adjacent_vertices< I > >.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.261.1 Detailed Description

`template<typename I> struct mln::morpho::attribute::count_adjacent_vertices< I >`

Count_Adjacent_Vertices accumulator class. The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 83 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.261.2 Member Function Documentation

10.261.2.1 `template<typename I> void mln::morpho::attribute::count_adjacent_vertices< I >::init () [inline]`

Manipulators.

Definition at line 132 of file morpho/attribute/count_adjacent_vertices.hh.

10.261.2.2 `template<typename I> bool mln::morpho::attribute::count_adjacent_vertices< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Definition at line 185 of file morpho/attribute/count_adjacent_vertices.hh.

10.261.2.3 `void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.261.2.4 `void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.261.2.5 `template<typename I> unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 169 of file morpho/attribute/count_adjacent_vertices.hh.

10.262 mln::morpho::attribute::height< I > Struct Template Reference

Height accumulator class.

```
#include <height.hh>
```

Inherits `base< unsigned, height< I > >`.

Public Member Functions

- unsigned `base_level () const`

Get base & current level of the accumulator.

- bool `is_valid()` const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- unsigned `to_result()` const
Get the value of the accumulator.
- void `init()`
Manipulators.

10.262.1 Detailed Description

template<typename I> struct mln::morpho::attribute::height< I >

Height accumulator class. The parameter I is the image type on which the accumulator of pixels is built.
Definition at line 80 of file morpho/attribute/height.hh.

10.262.2 Member Function Documentation

10.262.2.1 template<typename I> unsigned mln::morpho::attribute::height< I >::base_level () const [inline]

Get base & current level of the accumulator.
Definition at line 214 of file morpho/attribute/height.hh.

10.262.2.2 template<typename I> void mln::morpho::attribute::height< I >::init () [inline]

Manipulators.
Definition at line 131 of file morpho/attribute/height.hh.

10.262.2.3 template<typename I> bool mln::morpho::attribute::height< I >::is_valid () const [inline]

Check whether this accu is able to return a result.
Always true here.
Definition at line 231 of file morpho/attribute/height.hh.
Referenced by `mln::morpho::attribute::height< I >::to_result()`.

10.262.2.4 void mln::Accumulator< height< I > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.262.2.5 void mln::Accumulator< height< I > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.262.2.6 template<typename I> unsigned mln::morpho::attribute::height< I >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 204 of file morpho/attribute/height.hh.

References mln::morpho::attribute::height< I >::is_valid().

10.263 mln::morpho::attribute::sharpness< I > Struct Template Reference

Sharpness accumulator class.

```
#include <sharpness.hh>
```

Inherits base< double, sharpness< I > >.

Public Member Functions

- unsigned [area](#) () const
Give the area of the component.
- unsigned [height](#) () const
Give the height.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- double [to_result](#) () const
Get the value of the accumulator.

- unsigned `volume()` const

Give the volume of the component.

- void `init()`

Manipulators.

10.263.1 Detailed Description

template<typename I> struct mln::morpho::attribute::sharpness< I >

Sharpness accumulator class. The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 80 of file `sharpness.hh`.

10.263.2 Member Function Documentation

**10.263.2.1 template<typename I> unsigned mln::morpho::attribute::sharpness< I >::area ()
 const [inline]**

Give the area of the component.

Definition at line 190 of file `sharpness.hh`.

**10.263.2.2 template<typename I> unsigned mln::morpho::attribute::sharpness< I >::height ()
 const [inline]**

Give the height.

Definition at line 206 of file `sharpness.hh`.

**10.263.2.3 template<typename I> void mln::morpho::attribute::sharpness< I >::init ()
 [inline]**

Manipulators.

Definition at line 134 of file `sharpness.hh`.

**10.263.2.4 template<typename I> bool mln::morpho::attribute::sharpness< I >::is_valid ()
 const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 214 of file `sharpness.hh`.

10.263.2.5 void mln::Accumulator< sharpness< I > >::take_as_init (const T & t)
[**inherited**]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.263.2.6 void mln::Accumulator< sharpness< I > >::take_n_times (unsigned n, const T & t)
[**inherited**]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.263.2.7 template<typename I > double mln::morpho::attribute::sharpness< I >::to_result () const [**inline**]

Get the value of the accumulator.

Definition at line 175 of file sharpness.hh.

10.263.2.8 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::volume () const [**inline**]

Give the volume of the component.

Definition at line 198 of file sharpness.hh.

10.264 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits base< S, sum< I, S > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const argument &v)
Set the return value of the accumulator.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .

- S [to_result](#) () const
Get the value of the accumulator.
- void [untake](#) (const argument &v)
Untake a value from the accumulator.
- void [init](#) ()
Manipulators.

10.264.1 Detailed Description

```
template<typename I, typename S = typename mln::value::props< typename I ::value >::sum>
class mln::morpho::attribute::sum< I, S >
```

Suminality accumulator class.

Definition at line 80 of file morpho/attribute/sum.hh.

10.264.2 Member Function Documentation

10.264.2.1 `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::init () [inline]`

Manipulators.

Definition at line 137 of file morpho/attribute/sum.hh.

References `mln::literal::zero`.

10.264.2.2 `template<typename I , typename S > bool mln::morpho::attribute::sum< I, S >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Return always true.

Definition at line 229 of file morpho/attribute/sum.hh.

10.264.2.3 `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::set_value (const argument & v) [inline]`

Set the return value of the accumulator.

Definition at line 205 of file morpho/attribute/sum.hh.

10.264.2.4 `void mln::Accumulator< sum< I, S > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.264.2.5 `void mln::Accumulator< sum< I, S > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.264.2.6 `template<typename I , typename S > S mln::morpho::attribute::sum< I, S`
`>::to_result () const` **[inline]**

Get the value of the accumulator.

Definition at line 221 of file `morpho/attribute/sum.hh`.

10.264.2.7 `template<typename I , typename S > void mln::morpho::attribute::sum< I, S`
`>::untake (const argument & v)` **[inline]**

Untake a value from the accumulator.

Definition at line 189 of file `morpho/attribute/sum.hh`.

10.265 mln::morpho::attribute::volume< I > Struct Template Reference

Volume accumulator class.

`#include <volume.hh>`

Inherits `base< unsigned, volume< I > >`.

Public Member Functions

- unsigned `area` () const
Give the area.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.265.1 Detailed Description

template<typename I> struct mln::morpho::attribute::volume< I >

Volume accumulator class. The parameter I is the image type on which the accumulator of pixels is built.
Definition at line 78 of file morpho/attribute/volume.hh.

10.265.2 Member Function Documentation

**10.265.2.1 template<typename I> unsigned mln::morpho::attribute::volume< I >::area ()
const [inline]**

Give the area.

Definition at line 202 of file morpho/attribute/volume.hh.

**10.265.2.2 template<typename I> void mln::morpho::attribute::volume< I >::init ()
[inline]**

Manipulators.

Definition at line 130 of file morpho/attribute/volume.hh.

**10.265.2.3 template<typename I> bool mln::morpho::attribute::volume< I >::is_valid ()
const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 210 of file morpho/attribute/volume.hh.

**10.265.2.4 void mln::Accumulator< volume< I > >::take_as_init (const T & t)
[inherited]**

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

**10.265.2.5 void mln::Accumulator< volume< I > >::take_n_times (unsigned n, const T & t)
[inherited]**

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.265.2.6 template<typename I> unsigned mln::morpho::attribute::volume< I >::to_result () const [inline]

Get the value of the accumulator.

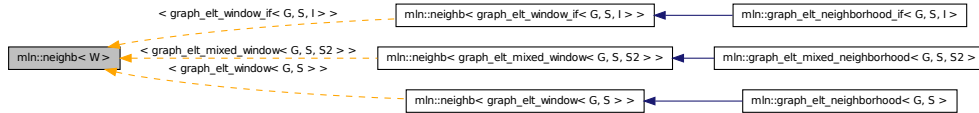
Definition at line 194 of file morpho/attribute/volume.hh.

10.266 mln::neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <neighb.hh>
```

Inheritance diagram for mln::neighb< W >:



Public Types

- `typedef neighb_bkd_niter< W > bkd_niter`
Backward site iterator associated type.
- `typedef neighb_fwd_niter< W > fwd_niter`
Forward site iterator associated type.
- `typedef fwd_niter niter`
Site iterator associated type.

Public Member Functions

- `neighb ()`
Constructor without argument.
- `neighb (const W &win)`
Constructor from a window win.

10.266.1 Detailed Description

```
template<typename W> class mln::neighb< W >
```

Adapter class from window to neighborhood.

Definition at line 76 of file `mln/core/neighb.hh`.

10.266.2 Member Typedef Documentation

10.266.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.266.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.266.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.266.3 Constructor & Destructor Documentation

10.266.3.1 `template<typename W > mln::neighb< W >::neighb () [inline]`

Constructor without argument.

Definition at line 150 of file mln/core/neighb.hh.

10.266.3.2 `template<typename W> mln::neighb< W >::neighb (const W & win) [inline]`

Constructor from a window `win`.

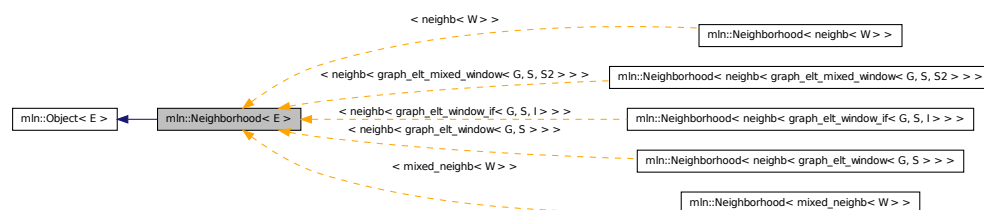
Definition at line 156 of file mln/core/neighb.hh.

10.267 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

```
#include <neighborhood.hh>
```

Inheritance diagram for `mln::Neighborhood< E >`:



10.267.1 Detailed Description

template<typename E> struct mln::Neighborhood< E >

Base class for implementation classes that are neighborhoods.

See also

[mln::doc::Neighborhood](#) for a complete documentation of this class contents.

Definition at line 66 of file core/concept/neighborhood.hh.

10.268 mln::Neighborhood< void > Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

10.268.1 Detailed Description

template<> struct mln::Neighborhood< void >

[Neighborhood](#) category flag type.

Definition at line 54 of file core/concept/neighborhood.hh.

10.269 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```

Inherited by [mln::Base< E >](#), [mln::Browsing< E >](#), [mln::Delta_Point_Site< E >](#), [mln::Function< E >](#), [mln::Gdpoint< E >](#), [mln::Graph< E >](#), [mln::Image< E >](#), [mln::io::off::internal::off_loader< I, E >](#), [mln::io::off::internal::off_saver< I, E >](#), [mln::Iterator< E >](#), [mln::Literal< E >](#), [mln::Mesh< E >](#), [mln::Meta_Accumulator< E >](#), [mln::Meta_Function< E >](#), [mln::Neighborhood< E >](#), [mln::Point_Site< E >](#), [mln::Proxy< E >](#), [mln::Site< E >](#), [mln::Site_Set< E >](#), [mln::Value< E >](#), [mln::value::HSL< E >](#), [mln::Value_Set< E >](#), [mln::Weighted_Window< E >](#), and [mln::Window< E >](#).

10.269.1 Detailed Description

template<typename E> struct mln::Object< E >

Base class for almost every class defined in Milena. The parameter *E* is the exact type.

Definition at line 171 of file object.hh.

10.270 mln::p2p_image< I, F > Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits `image_domain_morpher< I, I::domain_t, p2p_image< I, F > >`.

Public Types

- typedef `p2p_image< tag::image_< I >, tag::function_< F > >` [skeleton](#)
Skeleton.

Public Member Functions

- const `I::domain_t &` [domain](#) () const
Give the definition domain.
- const `F &` [fun](#) () const
Give the p2p function.
- `I::rvalue` [operator\(\)](#) (const typename `I::psite &p`) const
Read-only access to the image value located at point `p`.
- `internal::morpher_lvalue_< I >::ret` [operator\(\)](#) (const typename `I::psite &p`)
Read-write access to the image value located at point `p`.
- [p2p_image](#) (`I &ima`, const `F &f`)
Constructor from an image `ima` and a predicate `f`.
- [p2p_image](#) ()
Constructor without argument.

10.270.1 Detailed Description

```
template<typename I, typename F> struct mln::p2p_image< I, F >
```

FIXME: Doc!

Definition at line 90 of file `p2p_image.hh`.

10.270.2 Member Typedef Documentation

10.270.2.1 `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

Definition at line 95 of file `p2p_image.hh`.

10.270.3 Constructor & Destructor Documentation

10.270.3.1 `template<typename I , typename F > mln::p2p_image< I, F >::p2p_image ()
[inline]`

Constructor without argument.

Definition at line 178 of file p2p_image.hh.

10.270.3.2 `template<typename I , typename F > mln::p2p_image< I, F >::p2p_image (I & ima,
const F & f) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 184 of file p2p_image.hh.

10.270.4 Member Function Documentation

10.270.4.1 `template<typename I , typename F > const I::domain_t & mln::p2p_image< I, F
>::domain () const [inline]`

Give the definition domain.

Definition at line 201 of file p2p_image.hh.

10.270.4.2 `template<typename I , typename F > const F & mln::p2p_image< I, F >::fun ()
const [inline]`

Give the p2p function.

Definition at line 210 of file p2p_image.hh.

10.270.4.3 `template<typename I , typename F > I::rvalue mln::p2p_image< I, F >::operator() (
const typename I::psite & p) const [inline]`

Read-only access to the image value located at point *p*.

Definition at line 219 of file p2p_image.hh.

10.270.4.4 `template<typename I , typename F > internal::morpher_lvalue< I >::ret
mln::p2p_image< I, F >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at point *p*.

Definition at line 229 of file p2p_image.hh.

10.271 mln::p_array< P > Class Template Reference

Multi-set of sites.

```
#include <p_array.hh>
```

Inherits site_set_base< P, p_array< P > >.

Public Types

- typedef [p_indexed_bkd_piter](#)< [self_](#) > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef P [element](#)
Element associated type.
- typedef [p_indexed_fwd_piter](#)< [self_](#) > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef P [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_indexed_psite](#)< [self_](#) > [psite](#)
Psite associated type.

Public Member Functions

- [p_array](#)< P > & [append](#) (const P &p)
Append a point [p](#).
- [p_array](#)< P > & [append](#) (const [p_array](#)< P > &other)
Append an array [other](#) of points.
- void [change](#) (const [psite](#) &p, const P &new_p)
Change site [p](#) into [new_p](#).
- void [clear](#) ()
Clear this set.
- bool [has](#) (const util::index &i) const
Test is index [i](#) belongs to this site set.
- bool [has](#) (const [psite](#) &p) const
Test is [p](#) belongs to this site set.
- void [insert](#) (const P &p)
Insert a point [p](#) (equivalent as 'append').
- bool [is_valid](#) () const
Test this set validity so returns always true.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.

- unsigned [nsites](#) () const
Give the number of sites.
- const P & [operator\[\]](#) (const util::index &i) const
*Return the *i*-th element.*
- P & [operator\[\]](#) (unsigned i)
*Return the *i*-th site (mutable).*
- const P & [operator\[\]](#) (unsigned i) const
*Return the *i*-th site (constant).*
- [p_array](#) ()
Constructor.
- [p_array](#) (const std::vector< P > &vect)
*Constructor from a vector *vect*.*
- void [reserve](#) (size_type n)
*Reserve *n* cells.*
- void [resize](#) (size_t size)
Update the size of this array.
- const std::vector< P > & [std_vector](#) () const
Return the corresponding std::vector of points.

10.271.1 Detailed Description

template<typename P> class mln::p_array< P >

Multi-set of sites. [Site](#) set class based on std::vector.

Definition at line 84 of file p_array.hh.

10.271.2 Member Typedef Documentation

10.271.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 100 of file p_array.hh.

10.271.2.2 template<typename P> typedef P mln::p_array< P >::element

Element associated type.

Definition at line 91 of file p_array.hh.

10.271.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 97 of file p_array.hh.

10.271.2.4 `template<typename P> typedef P mln::p_array< P >::i_element`

Insertion element associated type.

Definition at line 141 of file p_array.hh.

10.271.2.5 `template<typename P> typedef fwd_piter mln::p_array< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 103 of file p_array.hh.

10.271.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite`

Psite associated type.

Definition at line 94 of file p_array.hh.

10.271.3 Constructor & Destructor Documentation

10.271.3.1 `template<typename P> mln::p_array< P >::p_array () [inline]`

Constructor.

Definition at line 340 of file p_array.hh.

10.271.3.2 `template<typename P> mln::p_array< P >::p_array (const std::vector< P > & vect) [inline]`

Constructor from a vector `vect`.

Definition at line 346 of file p_array.hh.

10.271.4 Member Function Documentation

10.271.4.1 `template<typename P> p_array< P > & mln::p_array< P >::append (const P & p) [inline]`

Append a point `p`.

Definition at line 408 of file p_array.hh.

Referenced by `mln::convert::to_p_array()`.

10.271.4.2 `template<typename P> p_array< P> & mln::p_array< P>::append (const p_array< P> & other) [inline]`

Append an array `other` of points.

Definition at line 425 of file `p_array.hh`.

References `mln::p_array< P>::std_vector()`.

10.271.4.3 `template<typename P> void mln::p_array< P>::change (const psite & p, const P & new_p) [inline]`

Change site `p` into `new_p`.

Definition at line 472 of file `p_array.hh`.

References `mln::p_array< P>::has()`.

10.271.4.4 `template<typename P> void mln::p_array< P>::clear () [inline]`

Clear this set.

Definition at line 436 of file `p_array.hh`.

10.271.4.5 `template<typename P> bool mln::p_array< P>::has (const psite & p) const [inline]`

Test is `p` belongs to this site set.

Definition at line 362 of file `p_array.hh`.

Referenced by `mln::p_array< P>::change()`, and `mln::p_array< P>::operator[]()`.

10.271.4.6 `template<typename P> bool mln::p_array< P>::has (const util::index & i) const [inline]`

Test is index `i` belongs to this site set.

Definition at line 375 of file `p_array.hh`.

References `mln::p_array< P>::nsites()`.

10.271.4.7 `template<typename P> void mln::p_array< P>::insert (const P & p) [inline]`

Insert a point `p` (equivalent as 'append').

Definition at line 417 of file `p_array.hh`.

10.271.4.8 `template<typename P> bool mln::p_array< P>::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 383 of file `p_array.hh`.

10.271.4.9 `template<typename P> std::size_t mln::p_array<P>::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 481 of file p_array.hh.

References `mln::p_array<P>::nsites()`.

10.271.4.10 `template<typename P> unsigned mln::p_array<P>::nsites () const [inline]`

Give the number of sites.

Definition at line 400 of file p_array.hh.

Referenced by `mln::registration::get_rot()`, `mln::p_array<P>::has()`, `mln::p_array<P>::memory_size()`, and `mln::p_array<P>::operator[]()`.

10.271.4.11 `template<typename P> P & mln::p_array<P>::operator[] (unsigned i) [inline]`

Return the `i-th` site (mutable).

Definition at line 463 of file p_array.hh.

References `mln::p_array<P>::nsites()`.

10.271.4.12 `template<typename P> const P & mln::p_array<P>::operator[] (const util::index & i) const [inline]`

Return the `i-th` element.

Definition at line 391 of file p_array.hh.

References `mln::p_array<P>::has()`.

10.271.4.13 `template<typename P> const P & mln::p_array<P>::operator[] (unsigned i) const [inline]`

Return the `i-th` site (constant).

Definition at line 454 of file p_array.hh.

References `mln::p_array<P>::nsites()`.

10.271.4.14 `template<typename P> void mln::p_array<P>::reserve (size_type n) [inline]`

Reserve `n` cells.

Definition at line 354 of file p_array.hh.

Referenced by `mln::convert::to_p_array()`.

10.271.4.15 `template<typename P> void mln::p_array< P >::resize (size_t size) [inline]`

Update the size of this array.

Definition at line 445 of file p_array.hh.

10.271.4.16 `template<typename P> const std::vector< P > & mln::p_array< P >::std_vector () const [inline]`

Return the corresponding std::vector of points.

Definition at line 489 of file p_array.hh.

Referenced by mln::p_array< P >::append().

10.272 mln::p_centered< W > Class Template Reference

[Site](#) set corresponding to a window centered on a site.

```
#include <p_centered.hh>
```

Inherits `site_set_base< W::psite, p_centered< W > >`, and `mlc_is_aW`.

Public Types

- `typedef p_centered_piter< W > bkd_piter`
Backward [Site_Iterator](#) associated type.
- `typedef psite element`
Element associated type.
- `typedef p_centered_piter< W > fwd_piter`
Forward [Site_Iterator](#) associated type.
- `typedef fwd_piter piter`
[Site_Iterator](#) associated type.
- `typedef W::psite psite`
Psite associated type.
- `typedef W::site site`
[Site](#) associated type.

Public Member Functions

- `const W::psite & center () const`
Give the center of this site set.
- `template<typename P>
bool has (const P &p) const`

Test if p belongs to the box.

- `bool is_valid () const`

Test if this site set is initialized.

- `std::size_t memory_size () const`

Return the size of this site set in memory.

- `p_centered (const W &win, const typename W::psite &c)`

Constructor from a window win and a center c .

- `p_centered ()`

Constructor without argument.

- `const W & window () const`

Give the window this site set is defined upon.

10.272.1 Detailed Description

`template<typename W> class mln::p_centered< W >`

[Site](#) set corresponding to a window centered on a site.

Definition at line 77 of file `p_centered.hh`.

10.272.2 Member Typedef Documentation

10.272.2.1 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 97 of file `p_centered.hh`.

10.272.2.2 `template<typename W> typedef psite mln::p_centered< W >::element`

Element associated type.

Definition at line 90 of file `p_centered.hh`.

10.272.2.3 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file `p_centered.hh`.

10.272.2.4 template<typename W> typedef fwd_piter mln::p_centered< W >::piter

[Site_Iterator](#) associated type.

Definition at line 100 of file p_centered.hh.

10.272.2.5 template<typename W> typedef W ::psite mln::p_centered< W >::psite

Psite associated type.

Definition at line 83 of file p_centered.hh.

10.272.2.6 template<typename W> typedef W ::site mln::p_centered< W >::site

[Site](#) associated type.

Definition at line 86 of file p_centered.hh.

10.272.3 Constructor & Destructor Documentation**10.272.3.1 template<typename W > mln::p_centered< W >::p_centered () [inline]**

Constructor without argument.

Definition at line 182 of file p_centered.hh.

10.272.3.2 template<typename W > mln::p_centered< W >::p_centered (const W & win, const typename W::psite & c) [inline]

Constructor from a window `win` and a center `c`.

Definition at line 188 of file p_centered.hh.

References `mln::p_centered< W >::is_valid()`.

10.272.4 Member Function Documentation**10.272.4.1 template<typename W > const W::psite & mln::p_centered< W >::center () const [inline]**

Give the center of this site set.

Definition at line 216 of file p_centered.hh.

10.272.4.2 template<typename W > template<typename P > bool mln::p_centered< W >::has (const P & p) const [inline]

Test if `p` belongs to the box.

Definition at line 199 of file p_centered.hh.

References `mln::p_centered< W >::is_valid()`.

10.272.4.3 `template<typename W> bool mln::p_centered<W>::is_valid () const [inline]`

Test if this site set is initialized.

Definition at line 175 of file `p_centered.hh`.

Referenced by `mln::p_centered<W>::has()`, and `mln::p_centered<W>::p_centered()`.

10.272.4.4 `template<typename W> std::size_t mln::p_centered<W>::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 208 of file `p_centered.hh`.

10.272.4.5 `template<typename W> const W & mln::p_centered<W>::window () const [inline]`

Give the window this site set is defined upon.

Definition at line 224 of file `p_centered.hh`.

10.273 `mln::p_complex<D, G>` Class Template Reference

A complex psite set based on the N-faces of a complex of dimension D (a D-complex).

`#include <p_complex.hh>`

Inherits `site_set_base<complex_psite<D, G>, p_complex<D, G>>`.

Public Types

- typedef `super_::site element`
Associated types.
- typedef `complex_psite<D, G> psite`
Point_Site associated type.
- typedef `p_complex_fwd_piter<D, G> fwd_piter`
Forward Site_Iterator associated type.
- typedef `p_complex_bkd_piter<D, G> bkd_piter`
Backward Site_Iterator associated type.
- typedef `fwd_piter piter`
Site_Iterator associated type.

Public Member Functions

- bool `has (const psite &p) const`
Does this site set has p?

- bool [is_valid](#) () const
Is this site set valid?
- unsigned [nfaces](#) () const
Return the number of faces in the complex.
- unsigned [nfaces_of_dim](#) (unsigned n) const
Return the number of n-faces in the complex.
- unsigned [nsites](#) () const
Return The number of sites of the set, i.e., the number of faces.
- [p_complex](#) (const [topo::complex](#)< D > &cplx, const G &geom)
Construct a complex psite set from a complex.
- [topo::complex](#)< D > & [cplx](#) () const
Accessors.
- [topo::complex](#)< D > & [cplx](#) ()
Return the complex associated to the [p_complex](#) domain (mutable version).
- const G & [geom](#) () const
Return the geometry of the complex.

10.273.1 Detailed Description

template<unsigned D, typename G> class mln::p_complex< D, G >

A complex psite set based on the N-faces of a complex of dimension D (a D-complex).

Template Parameters

D The dimension of the complex.

G A function object type, associating localization information (geometry) to each face of the complex.

See also

[mln::geom::complex_geometry](#). A complex [psite set](#) based on the N-faces of a complex.

Definition at line 116 of file p_complex.hh.

10.273.2 Member Typedef Documentation

**10.273.2.1 template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G>
mln::p_complex< D, G >::bkd_piter**

Backward [Site_Iterator](#) associated type.

Definition at line 141 of file p_complex.hh.

10.273.2.2 `template<unsigned D, typename G> typedef super_ ::site mln::p_complex< D, G >::element`

Associated types.

Element associated type.

Definition at line 132 of file p_complex.hh.

10.273.2.3 `template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G> mln::p_complex< D, G >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 138 of file p_complex.hh.

10.273.2.4 `template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G >::piter`

[Site_Iterator](#) associated type.

Definition at line 144 of file p_complex.hh.

10.273.2.5 `template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex< D, G >::psite`

Point_Site associated type.

Definition at line 135 of file p_complex.hh.

10.273.3 Constructor & Destructor Documentation

10.273.3.1 `template<unsigned D, typename G > mln::p_complex< D, G >::p_complex (const topo::complex< D > & cplx, const G & geom) [inline]`

Construct a complex psite set from a complex.

Parameters

cplx The complex upon which the complex psite set is built.

geom FIXME

Definition at line 231 of file p_complex.hh.

10.273.4 Member Function Documentation

10.273.4.1 `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx () const`

Accessors.

Return the complex associated to the [p_complex](#) domain (const version)

Definition at line 293 of file p_complex.hh.

References mln::p_complex< D, G >::is_valid().

Referenced by mln::complex_psite< D, G >::change_target(), mln::complex_psite< D, G >::complex_psite(), and mln::operator==().

10.273.4.2 `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx ()`

Return the complex associated to the [p_complex](#) domain (mutable version).

Definition at line 301 of file p_complex.hh.

References mln::p_complex< D, G >::is_valid().

10.273.4.3 `template<unsigned D, typename G > const G & mln::p_complex< D, G >::geom () const`

Return the geometry of the complex.

Definition at line 309 of file p_complex.hh.

10.273.4.4 `template<unsigned D, typename G > bool mln::p_complex< D, G >::has (const psite & p) const [inline]`

Does this site set has p ?

Definition at line 271 of file p_complex.hh.

References mln::complex_psite< D, G >::is_valid(), mln::p_complex< D, G >::is_valid(), and mln::complex_psite< D, G >::site_set().

10.273.4.5 `template<unsigned D, typename G > bool mln::p_complex< D, G >::is_valid () const [inline]`

Is this site set valid?

Definition at line 263 of file p_complex.hh.

Referenced by mln::p_complex< D, G >::cplx(), and mln::p_complex< D, G >::has().

10.273.4.6 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces () const [inline]`

Return the number of faces in the complex.

Definition at line 247 of file p_complex.hh.

Referenced by mln::p_complex< D, G >::nsites().

10.273.4.7 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces_of_dim (unsigned n) const [inline]`

Return the number of n -faces in the complex.

Definition at line 255 of file p_complex.hh.

10.273.4.8 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nsites ()
const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the [mln::Site_Set](#) concept, since the property trait::site_set::nsites::known of this site set is set to 'known'.)

Definition at line 239 of file p_complex.hh.

References `mln::p_complex< D, G >::nfaces()`.

10.274 mln::p_edges< G, F > Class Template Reference

[Site](#) set mapping graph edges and image sites.

```
#include <p_edges.hh>
```

Inherits `site_set_base_< F::result, p_edges< G, F > >`.

Public Types

- typedef [util::edge](#)< G > [edge](#)
Type of graph edge.
- typedef F [fun_t](#)
Function associated type.
- typedef [util::edge](#)< G > [graph_element](#)
Type of graph element this site set focuses on.
- typedef G [graph_t](#)
Graph associated type.
- typedef super_::site [element](#)
Associated types.
- typedef p_edges_psite< G, F > [psite](#)
Point_Site associated type.
- typedef [p_graph_piter](#)< [self_](#), mln_edge_fwd_iter(G) > [fwd_piter](#)
Forward Site_Iterator associated type.
- typedef [p_graph_piter](#)< [self_](#), mln_edge_bkd_iter(G) > [bkd_piter](#)
Backward Site_Iterator associated type.
- typedef [fwd_piter](#) [piter](#)
Site_Iterator associated type.

Public Member Functions

- `bool has (const psite &p) const`
Does this site set has site p?
- `template<typename G2 >
bool has (const util::edge< G2 > &e) const`
Does this site set has edge e?
- `void invalidate ()`
Invalidate this site set.
- `bool is_valid () const`
Is this site set valid?
- `std::size_t memory_size () const`
*Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>);
bool has(unsigned vertex_id) const;.*
- `unsigned nedges () const`
Return The number of edges in the graph.
- `unsigned nsites () const`
Return The number of points (sites) of the set, i.e., the number of edges.
- `p_edges ()`
*Constructors
Default constructor.*
- `p_edges (const Graph< G > &gr)`
Construct a graph edge psite set from a graph.
- `p_edges (const Graph< G > &gr, const Function< F > &f)`
Construct a graph edge psite set from a graph and a function.
- `template<typename F2 >
p_edges (const Graph< G > &gr, const Function< F2 > &f)`
Construct a graph edge psite set from a graph and a function.
- `const G & graph () const`
Accessors.
- `const F & function () const`
Return the mapping function.

10.274.1 Detailed Description

```
template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> class
mln::p_edges< G, F >
```

[Site](#) set mapping graph edges and image sites.

Definition at line 70 of file p_edges.hh.

10.274.2 Member Typedef Documentation

10.274.2.1 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef p_graph_piter< self_, mln_edge_bkd_iter(G) > mln::p_edges< G, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 128 of file p_edges.hh.

10.274.2.2 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef util::edge<G> mln::p_edges< G, F >::edge`

Type of graph edge.

Definition at line 86 of file p_edges.hh.

10.274.2.3 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef super_::site mln::p_edges< G, F >::element`

Associated types.

Element associated type.

Definition at line 119 of file p_edges.hh.

10.274.2.4 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef F mln::p_edges< G, F >::fun_t`

[Function](#) associated type.

Definition at line 83 of file p_edges.hh.

10.274.2.5 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef p_graph_piter< self_, mln_edge_fwd_iter(G) > mln::p_edges< G, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 125 of file p_edges.hh.

10.274.2.6 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
 typedef util::edge<G> mln::p_edges< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 89 of file p_edges.hh.

10.274.2.7 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
 typedef G mln::p_edges< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 80 of file p_edges.hh.

10.274.2.8 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
 typedef fwd_piter mln::p_edges< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 131 of file p_edges.hh.

10.274.2.9 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
 typedef p_edges_psite<G, F> mln::p_edges< G, F >::psite`

[Point_Site](#) associated type.

Definition at line 122 of file p_edges.hh.

10.274.3 Constructor & Destructor Documentation

10.274.3.1 `template<typename G , typename F > mln::p_edges< G, F >::p_edges ()
 [inline]`

Constructors

Default constructor.

Definition at line 203 of file p_edges.hh.

10.274.3.2 `template<typename G , typename F > mln::p_edges< G, F >::p_edges (const
 Graph< G > & gr) [inline]`

Construct a graph edge psite set from a graph.

Parameters

gr The graph upon which the graph edge psite set is built.

Definition at line 209 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

10.274.3.3 `template<typename G , typename F > mln::p_edges< G, F >::p_edges (const Graph< G > & gr, const Function< F > & f) [inline]`

Construct a graph edge psite set from a graph and a function.

Parameters

gr The graph upon which the graph edge psite set is built.

f the function mapping edges and sites.

Definition at line 221 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.274.3.4 `template<typename G , typename F > template<typename F2 > mln::p_edges< G, F >::p_edges (const Graph< G > & gr, const Function< F2 > & f) [inline]`

Construct a graph edge psite set from a graph and a function.

Parameters

gr The graph upon which the graph edge psite set is built.

f the function mapping edges and sites. It must be convertible towards the function type F.

Definition at line 231 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.274.4 Member Function Documentation

10.274.4.1 `template<typename G , typename F > const F & mln::p_edges< G, F >::function () const [inline]`

Return the mapping function.

Definition at line 324 of file p_edges.hh.

10.274.4.2 `template<typename G , typename F > const G & mln::p_edges< G, F >::graph () const [inline]`

Accessors.

Return the graph associated to this site set

Definition at line 315 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

Referenced by mln::operator==().

10.274.4.3 `template<typename G , typename F > bool mln::p_edges< G, F >::has (const psite & p) const [inline]`

Does this site set has site *p*?

Definition at line 276 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.274.4.4 `template<typename G , typename F > template<typename G2 > bool mln::p_edges< G, F >::has (const util::edge< G2 > & e) const [inline]`

Does this site set has edge *e*?

Definition at line 286 of file p_edges.hh.

References mln::util::edge< G >::graph(), mln::util::edge< G >::is_valid(), and mln::p_edges< G, F >::is_valid().

10.274.4.5 `template<typename G , typename F > void mln::p_edges< G, F >::invalidate () [inline]`

Invalidate this site set.

Definition at line 268 of file p_edges.hh.

10.274.4.6 `template<typename G , typename F > bool mln::p_edges< G, F >::is_valid () const [inline]`

Is this site set valid?

Definition at line 260 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::graph(), mln::p_edges< G, F >::has(), and mln::p_edges< G, F >::p_edges().

10.274.4.7 `template<typename G , typename F > std::size_t mln::p_edges< G, F >::memory_size () const [inline]`

Does this site set has *vertex_id*? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.

Definition at line 305 of file p_edges.hh.

10.274.4.8 `template<typename G , typename F > unsigned mln::p_edges< G, F >::nedges () const [inline]`

Return The number of edges in the graph.

Definition at line 252 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::nsites().

10.274.4.9 `template<typename G , typename F > unsigned mln::p_edges< G, F >::nsites () const [inline]`

Return The number of points (sites) of the set, i.e., the number of *edges*.

Definition at line 244 of file p_edges.hh.

References `mln::p_edges< G, F >::nedges()`.

10.275 `mln::p_faces< N, D, P >` Struct Template Reference

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

```
#include <p_faces.hh>
```

Inherits `site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >`.

Public Types

- typedef `super_::site` `element`
Associated types.
- typedef `faces_psite< N, D, P >` `psite`
Point_Site associated type.
- typedef `p_faces_fwd_piter_< N, D, P >` `fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `p_faces_bkd_piter_< N, D, P >` `bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `fwd_piter` `piter`
[Site_Iterator](#) associated type.

Public Member Functions

- `bool` `is_valid ()` `const`
Is this site set valid?
- `unsigned` `nfaces ()` `const`
Return The number of faces in the complex.
- `unsigned` `nsites ()` `const`
Return The number of sites of the set, i.e., the number of faces.
- `p_faces` (`const` `topo::complex< D >` &`cplx`)
Construct a faces psite set from an `mln::complex`.
- `p_faces` (`const` `p_complex< D, P >` &`pc`)
Construct a faces psite set from an `mln::p_complex`.
- `topo::complex< D >` &`cplx ()` `const`
Accessors.
- `topo::complex< D >` &`cplx ()`
Return the complex associated to the `p_faces` domain (mutable version).

10.275.1 Detailed Description

template<unsigned N, unsigned D, typename P> struct mln::p_faces< N, D, P >

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

Definition at line 77 of file p_faces.hh.

10.275.2 Member Typedef Documentation

10.275.2.1 template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 110 of file p_faces.hh.

10.275.2.2 template<unsigned N, unsigned D, typename P> typedef super_ ::site mln::p_faces< N, D, P >::element

Associated types.

Element associated type.

Definition at line 99 of file p_faces.hh.

10.275.2.3 template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 106 of file p_faces.hh.

10.275.2.4 template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter

[Site_Iterator](#) associated type.

Definition at line 113 of file p_faces.hh.

10.275.2.5 template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite

Point_Site associated type.

Definition at line 102 of file p_faces.hh.

10.275.3 Constructor & Destructor Documentation

10.275.3.1 template<unsigned N, unsigned D, typename P> mln::p_faces< N, D, P >::p_faces (const topo::complex< D > & cplx) [inline]

Construct a faces psite set from an mln::complex.

Parameters

cplx The complex upon which the complex psite set is built.

Definition at line 192 of file p_faces.hh.

10.275.3.2 `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces (const p_complex< D, P > & pc) [inline]`

Construct a faces psite set from an [mln::p_complex](#).

Parameters

pc The complex upon which the complex psite set is built.

Definition at line 201 of file p_faces.hh.

10.275.4 Member Function Documentation

10.275.4.1 `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx () const`

Accessors.

Return the complex associated to the [p_faces](#) domain (const version).

Definition at line 257 of file p_faces.hh.

References [mln::p_faces< N, D, P >::is_valid\(\)](#).

Referenced by [mln::operator==\(\)](#).

10.275.4.2 `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ()`

Return the complex associated to the [p_faces](#) domain (mutable version).

Definition at line 265 of file p_faces.hh.

References [mln::p_faces< N, D, P >::is_valid\(\)](#).

10.275.4.3 `template<unsigned N, unsigned D, typename P > bool mln::p_faces< N, D, P >::is_valid () const [inline]`

Is this site set valid?

Definition at line 227 of file p_faces.hh.

Referenced by [mln::p_faces< N, D, P >::cplx\(\)](#).

10.275.4.4 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nfaces () const [inline]`

Return The number of faces in the complex.

Definition at line 219 of file p_faces.hh.

Referenced by mln::p_faces< N, D, P >::nsites().

10.275.4.5 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nsites () const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the [mln::Site_Set](#) concept, since the property trait::site_set::nsites::known of this site set is set to 'known'.)

Definition at line 211 of file p_faces.hh.

References mln::p_faces< N, D, P >::nfaces().

10.276 mln::p_graph_piter< S, I > Class Template Reference

Generic iterator on point sites of a mln::S.

```
#include <p_graph_piter.hh>
```

Inherits [site_set_iterator_base< S, p_graph_piter< S, I > >](#).

Public Member Functions

- `const S::graph_t & graph () const`
Return the graph associated to the target S.
- `unsigned id () const`
Return the graph element id.
- `mln_q_subject (iter) element()`
Return the underlying graph element.
- `void next ()`
Go to the next element.
- `p_graph_piter ()`
Constructors.

10.276.1 Detailed Description

```
template<typename S, typename I> class mln::p_graph_piter< S, I >
```

Generic iterator on point sites of a mln::S.

Definition at line 55 of file p_graph_piter.hh.

10.276.2 Constructor & Destructor Documentation

10.276.2.1 `template<typename S , typename I > mln::p_graph_piter< S, I >::p_graph_piter ()
[inline]`

Constructors.

Definition at line 151 of file p_graph_piter.hh.

10.276.3 Member Function Documentation

10.276.3.1 `template<typename S , typename I > const S::graph_t & mln::p_graph_piter< S, I
>::graph () const [inline]`

Return the graph associated to the target S.

Definition at line 212 of file p_graph_piter.hh.

10.276.3.2 `template<typename S , typename I > unsigned mln::p_graph_piter< S, I >::id ()
const [inline]`

Return the graph element id.

Definition at line 228 of file p_graph_piter.hh.

10.276.3.3 `template<typename S , typename I > mln::p_graph_piter< S, I >::mln_q_subject (iter)`

Return the underlying graph element.

10.276.3.4 `void mln::Site_Iterator< p_graph_piter< S, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.277 mln::p_if< S, F > Class Template Reference

[Site](#) set restricted w.r.t.

```
#include <p_if.hh>
```

Inherits `site_set_base_< S::psite, p_if< S, F > >`.

Public Types

- typedef p_if_piter_< typename S::bkd_piter, S, F > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef S::element [element](#)
Element associated type.
- typedef p_if_piter_< typename S::fwd_piter, S, F > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef S::psite [psite](#)
Psite associated type.

Public Member Functions

- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the subset.
- bool [is_valid](#) () const
Test if this site set is valid.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- const S & [overset](#) () const
Give the primary overset.
- [p_if](#) ()
Constructor without argument.
- [p_if](#) (const S &s, const F &f)
Constructor with a site set s and a predicate f .
- bool [pred](#) (const [psite](#) &p) const
Test predicate on point site p .
- const F & [predicate](#) () const
Give the predicate function.

10.277.1 Detailed Description

template<typename S, typename F> class mln::p_if< S, F >

[Site](#) set restricted w.r.t. a predicate.

Parameter S is a site set type; parameter F is a function from point to Boolean.

Definition at line 83 of file p_if.hh.

10.277.2 Member Typedef Documentation

10.277.2.1 template<typename S, typename F> typedef p_if_piter_<typename S ::bkd_piter, S, F> mln::p_if< S, F >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 100 of file p_if.hh.

10.277.2.2 template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element

Element associated type.

Definition at line 90 of file p_if.hh.

10.277.2.3 template<typename S, typename F> typedef p_if_piter_<typename S ::fwd_piter, S, F> mln::p_if< S, F >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 97 of file p_if.hh.

10.277.2.4 template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter

[Site_Iterator](#) associated type.

Definition at line 103 of file p_if.hh.

10.277.2.5 template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite

Psite associated type.

Definition at line 94 of file p_if.hh.

10.277.3 Constructor & Destructor Documentation

10.277.3.1 template<typename S , typename F > mln::p_if< S, F >::p_if (const S & s, const F & f) [inline]

Constructor with a site set s and a predicate f.

Definition at line 190 of file p_if.hh.

Constructor without argument.

Definition at line 198 of file p_if.hh.

```
10.277.4.1 template<typename S , typename F > bool mln::p_if< S, F >::has ( const psite & p )
const [inline]
```

Test if p belongs to the subset.

Definition at line 159 of file p_if.hh.

References mln::p_if< S, F >::has().

Referenced by mln::p_if< S, F >::has().

Test if this site set is valid.

Definition at line 167 of file p_if.hh.

Return the size of this site set in memory.

Definition at line 213 of file p_if.hh.

Give the primary overset.

Definition at line 175 of file p_if.hh.

Test predicate on point site p .

Definition at line 183 of file p_if.hh.

Give the predicate function.

Definition at line 205 of file p_if.hh.

10.278 mln::p_image< I > Class Template Reference

[Site](#) set based on an image of Booleans.

```
#include <p_image.hh>
```

Inherits [site_set_base_< I::psite, p_image< I > >](#).

Public Types

- typedef [S::bkd_piter](#) [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef [I::psite](#) [element](#)
Element associated type.
- typedef [S::fwd_piter](#) [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [psite](#) [i_element](#)
Insertion element associated type.
- typedef [S::piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [I::psite](#) [psite](#)
Psite associated type.
- typedef [psite](#) [r_element](#)
Removal element associated type.
- typedef [internal::p_image_site_set< I >::ret](#) [S](#)
Equivalent [site_set](#) type.

Public Member Functions

- void [clear](#) ()
Clear this set.
- bool [has](#) (const [psite](#) &) const
*Test is the [psite](#) *p* belongs to this site set.*
- void [insert](#) (const [psite](#) &p)
*Insert a site *p*.*
- bool [is_valid](#) () const
Test if this site set is valid, i.e., initialized.
- [std::size_t](#) [memory_size](#) () const

Return the size of this site set in memory.

- unsigned [nsites](#) () const
Give the number of sites.
- operator typename internal::p_image_site_set< I >::ret () const
Conversion towards the equivalent site set.
- [p_image](#) ()
Constructor without argument.
- [p_image](#) (const I &ima)
Constructor.
- void [remove](#) (const [psite](#) &p)
Remove a site p.
- void [toggle](#) (const [psite](#) &p)
Change the status in/out of a site p.

10.278.1 Detailed Description

`template<typename I> class mln::p_image< I >`

[Site](#) set based on an image of Booleans.

Definition at line 88 of file `p_image.hh`.

10.278.2 Member Typedef Documentation

10.278.2.1 `template<typename I> typedef S ::bkd_piter mln::p_image< I >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 110 of file `p_image.hh`.

10.278.2.2 `template<typename I> typedef I ::psite mln::p_image< I >::element`

Element associated type.

Definition at line 100 of file `p_image.hh`.

10.278.2.3 `template<typename I> typedef S ::fwd_piter mln::p_image< I >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 107 of file `p_image.hh`.

10.278.2.4 template<typename I > typedef psite mln::p_image< I >::i_element

Insertion element associated type.

Definition at line 136 of file p_image.hh.

10.278.2.5 template<typename I > typedef S ::piter mln::p_image< I >::piter

[Site_Iterator](#) associated type.

Definition at line 113 of file p_image.hh.

10.278.2.6 template<typename I > typedef I ::psite mln::p_image< I >::psite

Psite associated type.

Definition at line 104 of file p_image.hh.

10.278.2.7 template<typename I > typedef psite mln::p_image< I >::r_element

Removal element associated type.

Definition at line 142 of file p_image.hh.

10.278.2.8 template<typename I > typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S

Equivalent site_set type.

Definition at line 93 of file p_image.hh.

10.278.3 Constructor & Destructor Documentation**10.278.3.1 template<typename I > mln::p_image< I >::p_image () [inline]**

Constructor without argument.

Definition at line 182 of file p_image.hh.

10.278.3.2 template<typename I > mln::p_image< I >::p_image (const I & ima) [inline]

Constructor.

Definition at line 189 of file p_image.hh.

References mln::p_image< I >::clear().

10.278.4 Member Function Documentation**10.278.4.1 template<typename I > void mln::p_image< I >::clear () [inline]**

Clear this set.

Definition at line 283 of file p_image.hh.

References mln::data::fill_with_value(), and mln::p_image< I >::is_valid().

Referenced by mln::p_image< I >::p_image().

10.278.4.2 **template<typename I > bool mln::p_image< I >::has (const psite & p) const [inline]**

Test is the psite p belongs to this site set.

Definition at line 199 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.278.4.3 **template<typename I > void mln::p_image< I >::insert (const psite & p) [inline]**

Insert a site p.

Definition at line 224 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.278.4.4 **template<typename I > bool mln::p_image< I >::is_valid () const [inline]**

Test if this site set is valid, i.e., initialized.

Definition at line 208 of file p_image.hh.

Referenced by mln::p_image< I >::clear(), mln::p_image< I >::has(), mln::p_image< I >::insert(), mln::p_image< I >::memory_size(), mln::p_image< I >::remove(), and mln::p_image< I >::toggle().

10.278.4.5 **template<typename I > std::size_t mln::p_image< I >::memory_size () const [inline]**

Return the size of this site set in memory.

Definition at line 273 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.278.4.6 **template<typename I > unsigned mln::p_image< I >::nsites () const [inline]**

Give the number of sites.

Definition at line 216 of file p_image.hh.

10.278.4.7 **template<typename I > mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret () const [inline]**

Conversion towards the equivalent site set.

Definition at line 174 of file p_image.hh.

10.278.4.8 `template<typename I> void mln::p_image< I >::remove (const psite & p)` `[inline]`

Remove a site `p`.

Definition at line 237 of file `p_image.hh`.

References `mln::p_image< I >::is_valid()`.

10.278.4.9 `template<typename I> void mln::p_image< I >::toggle (const psite & p)` `[inline]`

Change the status in/out of a site `p`.

Definition at line 251 of file `p_image.hh`.

References `mln::p_image< I >::is_valid()`.

10.279 `mln::p_indexed_bkd_piter< S >` Class Template Reference

Backward iterator on sites of an indexed site set.

`#include <p_array.hh>`

Inherits `site_set_iterator_base< S, p_indexed_bkd_piter< S > >`.

Public Member Functions

- `int index () const`
Return the current index.
- `void next ()`
Go to the next element.
- `p_indexed_bkd_piter (const S &s)`
Constructor.
- `p_indexed_bkd_piter ()`
Constructor with no argument.

10.279.1 Detailed Description

`template<typename S> class mln::p_indexed_bkd_piter< S >`

Backward iterator on sites of an indexed site set.

Definition at line 276 of file `p_array.hh`.

10.279.2 Constructor & Destructor Documentation

10.279.2.1 `template<typename S> mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ()`
`[inline]`

Constructor with no argument.

Definition at line 686 of file p_array.hh.

10.279.2.2 `template<typename S> mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter (`
`const S & s) [inline]`

Constructor.

Definition at line 692 of file p_array.hh.

10.279.3 Member Function Documentation

10.279.3.1 `template<typename S> int mln::p_indexed_bkd_piter< S >::index () const`
`[inline]`

Return the current index.

Definition at line 733 of file p_array.hh.

10.279.3.2 `void mln::Site_Iterator< p_indexed_bkd_piter< S > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.280 mln::p_indexed_fwd_piter< S > Class Template Reference

Forward iterator on sites of an indexed site set.

`#include <p_array.hh>`

Inherits `site_set_iterator_base< S, p_indexed_fwd_piter< S > >`.

Public Member Functions

- `int index () const`
Return the current index.
- `void next ()`

Go to the next element.

- [p_indexed_fwd_piter](#) (const S &s)

Constructor.

- [p_indexed_fwd_piter](#) ()

Constructor with no argument.

10.280.1 Detailed Description

template<typename S> class mln::p_indexed_fwd_piter< S >

Forward iterator on sites of an indexed site set.

Definition at line 235 of file p_array.hh.

10.280.2 Constructor & Destructor Documentation

**10.280.2.1 template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ()
 [inline]**

Constructor with no argument.

Definition at line 629 of file p_array.hh.

**10.280.2.2 template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ()
 const S & s) [inline]**

Constructor.

Definition at line 635 of file p_array.hh.

10.280.3 Member Function Documentation

**10.280.3.1 template<typename S > int mln::p_indexed_fwd_piter< S >::index () const
 [inline]**

Return the current index.

Definition at line 676 of file p_array.hh.

10.280.3.2 void mln::Site_Iterator< p_indexed_fwd_piter< S > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.281 mln::p_indexed_psite< S > Class Template Reference

Psite class for indexed site sets such as [p_array](#).

```
#include <p_array.hh>
```

Inherits pseudo_site_base_< const S::element &, p_indexed_psite< S > >.

10.281.1 Detailed Description

```
template<typename S> class mln::p_indexed_psite< S >
```

Psite class for indexed site sets such as [p_array](#). .

Definition at line 183 of file p_array.hh.

10.282 mln::p_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits site_set_base_< P, p_key< K, P > >.

Public Types

- typedef p_double_piter< [self_](#), mln_bkd_eiter(util::set< K >), typename [p_set](#)< P >::bkd_piter > [bkd_piter](#)
Backward [Site Iterator](#) associated type.
- typedef P [element](#)
Element associated type.
- typedef p_double_piter< [self_](#), mln_fwd_eiter(util::set< K >), typename [p_set](#)< P >::fwd_piter > [fwd_piter](#)
Forward [Site Iterator](#) associated type.
- typedef std::pair< K, P > [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) piter
[Site Iterator](#) associated type.
- typedef p_double_psite< [self_](#), [p_set](#)< P > > [psite](#)
Psite associated type.
- typedef P [r_element](#)

Removal element associated type.

Public Member Functions

- void [change_key](#) (const K &k, const K &new_k)
*Change the key *k* into a new value *new_k*.*
- template<typename F >
void [change_keys](#) (const [Function_v2v](#)< F > &f)
*Change the keys by applying the function *f*.*
- void [clear](#) ()
Clear this site set.
- bool [exists_key](#) (const K &key) const
*Test if the *priority* exists.*
- bool [has](#) (const [psite](#) &) const
*Test is the *psite p* belongs to this site set.*
- bool [has](#) (const P &p) const
*Test is the *psite p* belongs to this site set.*
- void [insert](#) (const [i_element](#) &k_p)
*Insert a pair *k_p* (key *k*, site *p*).*
- void [insert](#) (const K &k, const P &p)
*Insert a pair (key *k*, site *p*).*
- bool [is_valid](#) () const
Test this set validity so returns always true.
- const K & [key](#) (const P &p) const
*Give the key associated with site *p*.*
- const [util::set](#)< K > & [keys](#) () const
Give the set of keys.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const [p_set](#)< P > & [operator\(\)](#) (const K &key) const
*Give the queue with the priority *priority*.*
- [p_key](#) ()
Constructor.

- void [remove](#) (const P &p)
Remove a site p.
- void [remove_key](#) (const K &k)
Remove all sites with key k.

10.282.1 Detailed Description

template<typename K, typename P> class mln::p_key< K, P >

Priority queue class.

Definition at line 72 of file p_key.hh.

10.282.2 Member Typedef Documentation

10.282.2.1 template<typename K , typename P > typedef p_double_piter<self_, mln_bkd_eiter(util::set<K>), typename p_set<P>::bkd_piter> mln::p_key< K, P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 93 of file p_key.hh.

10.282.2.2 template<typename K , typename P > typedef P mln::p_key< K, P >::element

Element associated type.

Definition at line 79 of file p_key.hh.

10.282.2.3 template<typename K , typename P > typedef p_double_piter<self_, mln_fwd_eiter(util::set<K>), typename p_set<P>::fwd_piter> mln::p_key< K, P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 88 of file p_key.hh.

10.282.2.4 template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element

Insertion element associated type.

Definition at line 118 of file p_key.hh.

10.282.2.5 template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter

[Site_Iterator](#) associated type.

Definition at line 96 of file p_key.hh.

10.282.2.6 `template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

Definition at line 83 of file p_key.hh.

10.282.2.7 `template<typename K , typename P > typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

Definition at line 132 of file p_key.hh.

10.282.3 Constructor & Destructor Documentation

10.282.3.1 `template<typename K , typename P > mln::p_key< K, P >::p_key () [inline]`

Constructor.

Definition at line 208 of file p_key.hh.

10.282.4 Member Function Documentation

10.282.4.1 `template<typename K , typename P > void mln::p_key< K, P >::change_key (const K & k, const K & new_k) [inline]`

Change the key *k* into a new value *new_k*.

Definition at line 382 of file p_key.hh.

References `mln::p_set< P >::nsites()`.

10.282.4.2 `template<typename K , typename P > template<typename F > void mln::p_key< K, P >::change_keys (const Function_v2v< F > & f) [inline]`

Change the keys by applying the function *f*.

Definition at line 428 of file p_key.hh.

References `mln::util::set< T >::insert()`.

10.282.4.3 `template<typename K , typename P > void mln::p_key< K, P >::clear () [inline]`

Clear this site set.

Definition at line 462 of file p_key.hh.

10.282.4.4 `template<typename K , typename P > bool mln::p_key< K, P >::exists_key (const K & key) const [inline]`

Test if the *priority* exists.

Definition at line 520 of file p_key.hh.

Referenced by mln::p_key< K, P >::operator()().

10.282.4.5 `template<typename K , typename P > bool mln::p_key< K, P >::has (const P & p)
const [inline]`

Test is the psite p belongs to this site set.

Definition at line 227 of file p_key.hh.

10.282.4.6 `template<typename K , typename P > bool mln::p_key< K, P >::has (const psite &
) const [inline]`

Test is the psite p belongs to this site set.

Definition at line 217 of file p_key.hh.

Referenced by mln::p_key< K, P >::insert().

10.282.4.7 `template<typename K , typename P > void mln::p_key< K, P >::insert (const
i_element & k_p) [inline]`

Insert a pair k_p (key k, site p).

Definition at line 301 of file p_key.hh.

10.282.4.8 `template<typename K , typename P > void mln::p_key< K, P >::insert (const K &
k, const P & p) [inline]`

Insert a pair (key k, site p).

Definition at line 268 of file p_key.hh.

References mln::p_key< K, P >::has().

10.282.4.9 `template<typename K , typename P > bool mln::p_key< K, P >::is_valid () const
[inline]`

Test this set validity so returns always true.

Definition at line 236 of file p_key.hh.

10.282.4.10 `template<typename K , typename P > const K & mln::p_key< K, P >::key (const P
& p) const [inline]`

Give the key associated with site p.

Definition at line 501 of file p_key.hh.

10.282.4.11 `template<typename K , typename P > const util::set< K > & mln::p_key< K, P
>::keys () const [inline]`

Give the set of keys.

Definition at line 511 of file p_key.hh.

10.282.4.12 `template<typename K , typename P > std::size_t mln::p_key< K, P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 475 of file p_key.hh.

10.282.4.13 `template<typename K , typename P > unsigned mln::p_key< K, P >::nsites () const [inline]`

Give the number of sites.

Definition at line 245 of file p_key.hh.

10.282.4.14 `template<typename K , typename P > const p_set< P > & mln::p_key< K, P >::operator() (const K & key) const [inline]`

Give the queue with the priority `priority`.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 489 of file p_key.hh.

References `mln::p_key< K, P >::exists_key()`.

10.282.4.15 `template<typename K , typename P > void mln::p_key< K, P >::remove (const P & p) [inline]`

Remove a site `p`.

Definition at line 309 of file p_key.hh.

10.282.4.16 `template<typename K , typename P > void mln::p_key< K, P >::remove_key (const K & k) [inline]`

Remove all sites with key `k`.

Definition at line 351 of file p_key.hh.

References `mln::p_set< P >::nsites()`.

10.283 mln::p_line2d Class Reference

2D discrete line of points.

`#include <p_line2d.hh>`

Inherits `site_set_base< point2d, p_line2d >`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`

Backward [Site_Iterator](#) associated type.

- typedef [point2d](#) [element](#)
Element associated type.
- typedef [p_indexed_fwd_piter](#)< [self_](#) > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [p_indexed_fwd_piter](#)< [self_](#) > [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_indexed_psite](#)< [self_](#) > [psite](#)
Psite associated type.
- typedef const [box2d](#) & [q_box](#)
[Box](#) (qualified) associated type.

Public Member Functions

- const [box2d](#) & [bbox](#) () const
Give the exact bounding box.
- const [point2d](#) & [begin](#) () const
Give the point that begins the line.
- const [point2d](#) & [end](#) () const
Give the point that ends the line.
- bool [has](#) (const util::index &i) const
*Test if index *i* belongs to this point set.*
- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this point set.*
- bool [is_valid](#) () const
Test if this line is valid, i.e., initialized.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of points.
- const [point2d](#) & [operator\[\]](#) (unsigned i) const
*Return the *i*-th point of the line.*
- [p_line2d](#) (const [point2d](#) &beg, const [point2d](#) &end, bool is_end_excluded=false)
*Constructor from point *beg* to point *end*.*

- [p_line2d](#) ()
Constructor without argument.
- `const std::vector< point2d > & std_vector () const`
Return the corresponding std::vector of points.

10.283.1 Detailed Description

2D discrete line of points. It is based on [p_array](#).

Definition at line 79 of file `p_line2d.hh`.

10.283.2 Member Typedef Documentation

10.283.2.1 `typedef p_indexed_bkd_piter<self_> mln::p_line2d::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 97 of file `p_line2d.hh`.

10.283.2.2 `typedef point2d mln::p_line2d::element`

Element associated type.

Definition at line 85 of file `p_line2d.hh`.

10.283.2.3 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file `p_line2d.hh`.

10.283.2.4 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter`

[Site_Iterator](#) associated type.

Definition at line 91 of file `p_line2d.hh`.

10.283.2.5 `typedef p_indexed_psite<self_> mln::p_line2d::psite`

Psite associated type.

Definition at line 88 of file `p_line2d.hh`.

10.283.2.6 `typedef const box2d& mln::p_line2d::q_box`

[Box](#) (qualified) associated type.

Definition at line 132 of file `p_line2d.hh`.

10.283.3 Constructor & Destructor Documentation

10.283.3.1 mln::p_line2d::p_line2d () [inline]

Constructor without argument.

Definition at line 161 of file p_line2d.hh.

References is_valid().

10.283.3.2 mln::p_line2d::p_line2d (const point2d & beg, const point2d & end, bool is_end_excluded = false) [inline]

Constructor from point beg to point end.

Definition at line 167 of file p_line2d.hh.

References is_valid().

10.283.4 Member Function Documentation

10.283.4.1 const box2d & mln::p_line2d::bbox () const [inline]

Give the exact bounding box.

Definition at line 273 of file p_line2d.hh.

References is_valid().

10.283.4.2 const point2d & mln::p_line2d::begin () const [inline]

Give the point that begins the line.

Definition at line 307 of file p_line2d.hh.

References is_valid().

Referenced by mln::debug::draw_graph().

10.283.4.3 const point2d & mln::p_line2d::end () const [inline]

Give the point that ends the line.

Definition at line 315 of file p_line2d.hh.

References is_valid(), and nsites().

Referenced by mln::debug::draw_graph().

10.283.4.4 bool mln::p_line2d::has (const psite & p) const [inline]

Test if p belongs to this point set.

Definition at line 240 of file p_line2d.hh.

10.283.4.5 `bool mln::p_line2d::has (const util::index & i) const [inline]`

Test if index `i` belongs to this point set.

Definition at line 251 of file `p_line2d.hh`.

References `nsites()`.

10.283.4.6 `bool mln::p_line2d::is_valid () const [inline]`

Test if this line is valid, i.e., initialized.

Definition at line 258 of file `p_line2d.hh`.

References `mln::implies()`.

Referenced by `bbox()`, `begin()`, `end()`, and `p_line2d()`.

10.283.4.7 `std::size_t mln::p_line2d::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 323 of file `p_line2d.hh`.

10.283.4.8 `unsigned mln::p_line2d::nsites () const [inline]`

Give the number of points.

Definition at line 266 of file `p_line2d.hh`.

Referenced by `end()`, `has()`, and `operator[]()`.

10.283.4.9 `const point2d & mln::p_line2d::operator[] (unsigned i) const [inline]`

Return the `i`-th point of the line.

Definition at line 299 of file `p_line2d.hh`.

References `nsites()`.

10.283.4.10 `const std::vector< point2d > & mln::p_line2d::std_vector () const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 281 of file `p_line2d.hh`.

10.284 `mln::p_mutable_array_of< S >` Class Template Reference

`p_mutable_array_of` is a mutable array of site sets.

`#include <p_mutable_array_of.hh>`

Inherits `site_set_base< S::site, p_mutable_array_of< S >>`.

Public Types

- typedef p_double_piter< [self_](#), mln_bkd_eiter([array_](#)), typename S::bkd_piter > [bkd_piter](#)
Backward [Site Iterator](#) associated type.
- typedef S [element](#)
Element associated type.
- typedef p_double_piter< [self_](#), mln_fwd_eiter([array_](#)), typename S::fwd_piter > [fwd_piter](#)
Forward [Site Iterator](#) associated type.
- typedef S [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) piter
[Site Iterator](#) associated type.
- typedef p_double_psite< [self_](#), [element](#) > [psite](#)
Psite associated type.

Public Member Functions

- void [clear](#) ()
Clear this set.
- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this point set.*
- void [insert](#) (const S &s)
*Insert a site set *s*.*
- bool [is_valid](#) () const
Test this set validity so returns always true.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nelements](#) () const
Give the number of elements (site sets) of this composite.
- S & [operator\[\]](#) (unsigned i)
*Return the *i*-th site set (mutable version).*
- const S & [operator\[\]](#) (unsigned i) const
*Return the *i*-th site set (const version).*
- [p_mutable_array_of](#) ()
Constructor without arguments.

- void [reserve](#) (unsigned n)
Reserve memory for n elements.

10.284.1 Detailed Description

template<typename S> class mln::p_mutable_array_of< S >

[p_mutable_array_of](#) is a mutable array of site sets. Parameter S is the type of the contained site sets.

Definition at line 76 of file p_mutable_array_of.hh.

10.284.2 Member Typedef Documentation

**10.284.2.1 template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(array_),
 typename S ::bkd_piter> mln::p_mutable_array_of< S >::bkd_piter**

Backward [Site_Iterator](#) associated type.

Definition at line 99 of file p_mutable_array_of.hh.

10.284.2.2 template<typename S > typedef S mln::p_mutable_array_of< S >::element

Element associated type.

Definition at line 85 of file p_mutable_array_of.hh.

**10.284.2.3 template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(array_),
 typename S ::fwd_piter> mln::p_mutable_array_of< S >::fwd_piter**

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file p_mutable_array_of.hh.

10.284.2.4 template<typename S > typedef S mln::p_mutable_array_of< S >::i_element

Insertion element associated type.

Definition at line 121 of file p_mutable_array_of.hh.

10.284.2.5 template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter

[Site_Iterator](#) associated type.

Definition at line 102 of file p_mutable_array_of.hh.

**10.284.2.6 template<typename S > typedef p_double_psite<self_, element>
 mln::p_mutable_array_of< S >::psite**

Psite associated type.

Definition at line 89 of file p_mutable_array_of.hh.

10.284.3 Constructor & Destructor Documentation

10.284.3.1 `template<typename S> mln::p_mutable_array_of< S >::p_mutable_array_of ()`
`[inline]`

Constructor without arguments.

Definition at line 175 of file p_mutable_array_of.hh.

10.284.4 Member Function Documentation

10.284.4.1 `template<typename S> void mln::p_mutable_array_of< S >::clear ()` `[inline]`

Clear this set.

Definition at line 241 of file p_mutable_array_of.hh.

10.284.4.2 `template<typename S> bool mln::p_mutable_array_of< S >::has (const psite & p)`
`const [inline]`

Test if *p* belongs to this point set.

Definition at line 190 of file p_mutable_array_of.hh.

10.284.4.3 `template<typename S> void mln::p_mutable_array_of< S >::insert (const S & s)`
`[inline]`

Insert a site set *s*.

Precondition

s is valid.

Definition at line 206 of file p_mutable_array_of.hh.

10.284.4.4 `template<typename S> bool mln::p_mutable_array_of< S >::is_valid () const`
`[inline]`

Test this set validity so returns always true.

Definition at line 198 of file p_mutable_array_of.hh.

10.284.4.5 `template<typename S> std::size_t mln::p_mutable_array_of< S >::memory_size ()`
`const [inline]`

Return the size of this site set in memory.

Definition at line 258 of file p_mutable_array_of.hh.

10.284.4.6 `template<typename S> unsigned mln::p_mutable_array_of< S >::nelements ()`
`const [inline]`

Give the number of elements (site sets) of this composite.

Definition at line 233 of file p_mutable_array_of.hh.

10.284.4.7 `template<typename S> S & mln::p_mutable_array_of< S >::operator[] (unsigned i) [inline]`

Return the `i`-th site set (mutable version).

Definition at line 224 of file p_mutable_array_of.hh.

10.284.4.8 `template<typename S> const S & mln::p_mutable_array_of< S >::operator[] (unsigned i) const [inline]`

Return the `i`-th site set (const version).

Definition at line 215 of file p_mutable_array_of.hh.

10.284.4.9 `template<typename S> void mln::p_mutable_array_of< S >::reserve (unsigned n) [inline]`

Reserve memory for `n` elements.

Definition at line 182 of file p_mutable_array_of.hh.

10.285 mln::p_n_faces_bkd_piter< D, G > Class Template Reference

Backward iterator on the `n`-faces sites of an `mln::p_complex<D, G>`.

`#include <p_n_faces_piter.hh>`

Inherits `p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >`.

Public Member Functions

- `void next ()`

Go to the next element.

- `p_n_faces_bkd_piter ()`

Construction and assignment.

- `unsigned n () const`

Accessors.

10.285.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_bkd_piter< D, G >`

Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

Definition at line 92 of file `p_n_faces_piter.hh`.

10.285.2 Constructor & Destructor Documentation

10.285.2.1 `template<unsigned D, typename G > mln::p_n_faces_bkd_piter< D, G >::p_n_faces_bkd_piter () [inline]`

Construction and assignment.

Definition at line 169 of file `p_n_faces_piter.hh`.

10.285.3 Member Function Documentation

10.285.3.1 `template<unsigned D, typename G > unsigned mln::p_n_faces_bkd_piter< D, G >::n () const [inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 186 of file `p_n_faces_piter.hh`.

10.285.3.2 `void mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.286 mln::p_n_faces_fwd_piter< D, G > Class Template Reference

Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

`#include <p_n_faces_piter.hh>`

Inherits `p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [p_n_faces_fwd_piter](#) ()
Construction and assignment.
- unsigned [n](#) () const
Accessors.

10.286.1 Detailed Description

template<unsigned D, typename G> class mln::p_n_faces_fwd_piter< D, G >

Forward iterator on the n-faces sites of an mln::p_complex<D, G>.

Definition at line 56 of file p_n_faces_piter.hh.

10.286.2 Constructor & Destructor Documentation

10.286.2.1 template<unsigned D, typename G > mln::p_n_faces_fwd_piter< D, G >::p_n_faces_fwd_piter () [inline]

Construction and assignment.

Definition at line 132 of file p_n_faces_piter.hh.

10.286.3 Member Function Documentation

10.286.3.1 template<unsigned D, typename G > unsigned mln::p_n_faces_fwd_piter< D, G >::n () const [inline]

Accessors.

Shortcuts to face_'s accessors.

Definition at line 149 of file p_n_faces_piter.hh.

10.286.3.2 void mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.287 mln::p_priority< P, Q > Class Template Reference

Priority queue.

```
#include <p_priority.hh>
```

Inherits `site_set_base_< Q::site, p_priority< P, Q > >`.

Public Types

- typedef `p_double_piter< self_, mln_fwd_eiter(util::set< P >), typename Q::bkd_piter > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `Q::element element`
Element associated type.
- typedef `p_double_piter< self_, mln_bkd_eiter(util::set< P >), typename Q::fwd_piter > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `std::pair< P, element > i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
[Site_Iterator](#) associated type.
- typedef `p_double_psite< self_, Q > psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear the queue.
- bool `exists_priority (const P &priority) const`
Test if the `priority` exists.
- const `Q::element & front () const`
Give an element with highest priority.
- bool `has (const psite &) const`
Test is the psite `p` belongs to this site set.
- const `P highest_priority () const`
Give the highest priority.
- void `insert (const i_element &p_e)`
Insert a pair `p_e` (priority `p`, element `e`).

- void [insert](#) (const [p_priority](#)< P, Q > &other)
Insert elements from another priority queue.
- bool [is_valid](#) () const
Test this set validity so returns always true.
- const P [lowest_priority](#) () const
Give the lowest priority.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const Q & [operator](#)() (const P &priority) const
*Give the queue with the priority *priority*.*
- [p_priority](#) ()
Constructor.
- void [pop](#) ()
Pop (remove) from the queue an element with highest priority.
- Q::element [pop_front](#) ()
Return an element with highest priority and remove it from the set.
- const [util::set](#)< P > & [priorities](#) () const
Give the set of priorities.
- void [push](#) (const P &priority, const [element](#) &e)
*Push in the queue with *priority* the element *e*.*

10.287.1 Detailed Description

template<typename P, typename Q> class mln::p_priority< P, Q >

Priority queue. The parameter P is the type of the priorities (for instance unsigned).

The parameter Q is a type of queue (for instance p_queue<point2d>).

Definition at line 76 of file p_priority.hh.

10.287.2 Member Typedef Documentation

10.287.2.1 **template<typename P, typename Q> typedef p_double_piter< self_,
mln_fwd_eiter(util::set<P>), typename Q ::bkd_piter > mln::p_priority< P, Q
>::bkd_piter**

Backward [Site_Iterator](#) associated type.

Definition at line 98 of file p_priority.hh.

10.287.2.2 `template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element`

Element associated type.

Definition at line 84 of file p_priority.hh.

10.287.2.3 `template<typename P, typename Q> typedef p_double_piter< self_, mln_bkd_eiter(util::set<P>), typename Q ::fwd_piter > mln::p_priority< P, Q >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 93 of file p_priority.hh.

10.287.2.4 `template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element`

Insertion element associated type.

Definition at line 121 of file p_priority.hh.

10.287.2.5 `template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter`

[Site_Iterator](#) associated type.

Definition at line 101 of file p_priority.hh.

10.287.2.6 `template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite`

Psite associated type.

Definition at line 88 of file p_priority.hh.

10.287.3 Constructor & Destructor Documentation

10.287.3.1 `template<typename P , typename Q > mln::p_priority< P, Q >::p_priority () [inline]`

Constructor.

Definition at line 202 of file p_priority.hh.

10.287.4 Member Function Documentation

10.287.4.1 `template<typename P , typename Q > void mln::p_priority< P, Q >::clear () [inline]`

Clear the queue.

Definition at line 316 of file p_priority.hh.

10.287.4.2 `template<typename P , typename Q > bool mln::p_priority< P, Q >::exists_priority (const P & priority) const [inline]`

Test if the `priority` exists.

Definition at line 366 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::operator()`.

10.287.4.3 `template<typename P , typename Q > const Q::element & mln::p_priority< P, Q >::front () const [inline]`

Give an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 294 of file p_priority.hh.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.287.4.4 `template<typename P , typename Q > bool mln::p_priority< P, Q >::has (const psite &) const [inline]`

Test is the `psite p` belongs to this site set.

Definition at line 211 of file p_priority.hh.

10.287.4.5 `template<typename P , typename Q > const P mln::p_priority< P, Q >::highest_priority () const [inline]`

Give the highest priority.

Precondition

`! is_empty()`

Definition at line 375 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::front()`, and `mln::p_priority< P, Q >::pop()`.

10.287.4.6 `template<typename P , typename Q > void mln::p_priority< P, Q >::insert (const i_element & p_e) [inline]`

Insert a pair `p_e` (priority `p`, element `e`).

Definition at line 251 of file p_priority.hh.

References `mln::p_priority< P, Q >::push()`.

10.287.4.7 `template<typename P , typename Q > void mln::p_priority< P, Q >::insert (const p_priority< P, Q > & other) [inline]`

Insert elements from another priority queue.

Definition at line 259 of file p_priority.hh.

10.287.4.8 `template<typename P , typename Q > bool mln::p_priority< P, Q >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 221 of file p_priority.hh.

10.287.4.9 `template<typename P , typename Q > const P mln::p_priority< P, Q >::lowest_priority () const [inline]`

Give the lowest priority.

Precondition

! is_empty()

Definition at line 384 of file p_priority.hh.

10.287.4.10 `template<typename P , typename Q > std::size_t mln::p_priority< P, Q >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 328 of file p_priority.hh.

10.287.4.11 `template<typename P , typename Q > unsigned mln::p_priority< P, Q >::nsites () const [inline]`

Give the number of sites.

Definition at line 230 of file p_priority.hh.

Referenced by mln::p_priority< P, Q >::operator()().

10.287.4.12 `template<typename P , typename Q > const Q & mln::p_priority< P, Q >::operator() (const P & priority) const [inline]`

Give the queue with the priority `priority`.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 341 of file p_priority.hh.

References mln::p_priority< P, Q >::exists_priority(), and mln::p_priority< P, Q >::nsites().

10.287.4.13 `template<typename P , typename Q > void mln::p_priority< P, Q >::pop ()`
[inline]

Pop (remove) from the queue an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 277 of file `p_priority.hh`.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.287.4.14 `template<typename P , typename Q > Q::element mln::p_priority< P, Q`
`>::pop_front () [inline]`

Return an element with highest priority and remove it from the set.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 304 of file `p_priority.hh`.

10.287.4.15 `template<typename P , typename Q > const util::set< P > & mln::p_priority< P, Q`
`>::priorities () const [inline]`

Give the set of priorities.

Definition at line 357 of file `p_priority.hh`.

10.287.4.16 `template<typename P , typename Q > void mln::p_priority< P, Q >::push (const P`
`& priority, const element & e) [inline]`

Push in the queue with `priority` the element `e`.

Definition at line 239 of file `p_priority.hh`.

Referenced by `mln::p_priority< P, Q >::insert()`, `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.288 `mln::p_queue< P >` Class Template Reference

Queue of sites (based on `std::deque`).

`#include <p_queue.hh>`

Inherits `site_set_base< P, p_queue< P > >`.

Public Types

- typedef [p_indexed_bkd_piter](#)< [self_](#) > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef P [element](#)
Element associated type.
- typedef [p_indexed_fwd_piter](#)< [self_](#) > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef P [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_indexed_psite](#)< [self_](#) > [psite](#)
Psite associated type.

Public Member Functions

- void [clear](#) ()
Clear the queue.
- const P & [front](#) () const
Give the front site p of the queue; p is the least recently inserted site.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this site set.
- bool [has](#) (const util::index &i) const
Test if index i belongs to this site set.
- void [insert](#) (const P &p)
Insert a site p (equivalent as 'push').
- bool [is_valid](#) () const
This set is always valid so it returns true.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const P & [operator\[\]](#) (unsigned i) const
Return the i -th site.

- [p_queue](#) ()
Constructor without argument.
- void [pop](#) ()
Pop (remove) the front site p from the queue; p is the least recently inserted site.
- [P pop_front](#) ()
Pop (remove) the front site p from the queue; p is the least recently inserted site and give the front site p of the queue; p is the least recently inserted site.
- void [push](#) (const P &p)
Push a site p in the queue.
- const std::deque< P > & [std_deque](#) () const
Return the corresponding std::deque of sites.

10.288.1 Detailed Description

template<typename P> class mln::p_queue< P >

Queue of sites (based on std::deque). The parameter P shall be a site or pseudo-site type.

Definition at line 74 of file p_queue.hh.

10.288.2 Member Typedef Documentation

10.288.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 90 of file p_queue.hh.

10.288.2.2 template<typename P> typedef P mln::p_queue< P >::element

Element associated type.

Definition at line 80 of file p_queue.hh.

10.288.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 87 of file p_queue.hh.

10.288.2.4 template<typename P> typedef P mln::p_queue< P >::i_element

Insertion element associated type.

Definition at line 118 of file p_queue.hh.

10.288.2.5 template<typename P> typedef fwd_piter mln::p_queue< P >::piter

[Site_Iterator](#) associated type.

Definition at line 93 of file p_queue.hh.

10.288.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite

Psite associated type.

Definition at line 84 of file p_queue.hh.

10.288.3 Constructor & Destructor Documentation**10.288.3.1 template<typename P> mln::p_queue< P >::p_queue () [inline]**

Constructor without argument.

Definition at line 163 of file p_queue.hh.

10.288.4 Member Function Documentation**10.288.4.1 template<typename P> void mln::p_queue< P >::clear () [inline]**

Clear the queue.

Definition at line 244 of file p_queue.hh.

10.288.4.2 template<typename P> const P & mln::p_queue< P >::front () const [inline]

Give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 224 of file p_queue.hh.

Referenced by mln::p_queue< P >::pop_front(), and mln::geom::impl::seeds2tiling().

10.288.4.3 template<typename P> bool mln::p_queue< P >::has (const util::index & *i*) const [inline]

Test if index *i* belongs to this site set.

Definition at line 183 of file p_queue.hh.

References mln::p_queue< P >::nsites().

10.288.4.4 template<typename P> bool mln::p_queue< P >::has (const psite & *p*) const [inline]

Test if *p* belongs to this site set.

Definition at line 170 of file p_queue.hh.

References mln::p_queue< P >::nsites().

10.288.4.5 template<typename P> void mln::p_queue< P >::insert (const P & p) [inline]

Insert a site *p* (equivalent as 'push').

Definition at line 261 of file `p_queue.hh`.

References `mln::p_queue< P >::push()`.

10.288.4.6 template<typename P> bool mln::p_queue< P >::is_valid () const [inline]

This set is always valid so it returns true.

Definition at line 191 of file `p_queue.hh`.

10.288.4.7 template<typename P> std::size_t mln::p_queue< P >::memory_size () const [inline]

Return the size of this site set in memory.

Definition at line 277 of file `p_queue.hh`.

References `mln::p_queue< P >::nsites()`.

10.288.4.8 template<typename P> unsigned mln::p_queue< P >::nsites () const [inline]

Give the number of sites.

Definition at line 199 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::has()`, `mln::p_queue< P >::memory_size()`, and `mln::p_queue< P >::operator[]()`.

10.288.4.9 template<typename P> const P & mln::p_queue< P >::operator[] (unsigned i) const [inline]

Return the *i*-th site.

Definition at line 252 of file `p_queue.hh`.

References `mln::p_queue< P >::nsites()`.

10.288.4.10 template<typename P> void mln::p_queue< P >::pop () [inline]

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.

Definition at line 215 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

10.288.4.11 template<typename P> P mln::p_queue< P >::pop_front () [inline]

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site and give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 233 of file `p_queue.hh`.

References mln::p_queue< P >::front(), and mln::p_queue< P >::pop().

10.288.4.12 `template<typename P> void mln::p_queue< P >::push (const P & p) [inline]`

Push a site `p` in the queue.

Definition at line 207 of file `p_queue.hh`.

Referenced by mln::p_queue< P >::insert(), and mln::geom::impl::seeds2tiling().

10.288.4.13 `template<typename P> const std::deque< P > & mln::p_queue< P >::std_deque () const [inline]`

Return the corresponding `std::deque` of sites.

Definition at line 269 of file `p_queue.hh`.

10.289 mln::p_queue_fast< P > Class Template Reference

Queue of sites class (based on [p_array](#)).

```
#include <p_queue_fast.hh>
```

Inherits `site_set_base_< P, p_queue_fast< P > >`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `P i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
[Site_Iterator](#) associated type.
- typedef `p_indexed_psite< self_ > psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear the queue.

- `bool compute_has (const P &p) const`
Test if p belongs to this site set.
- `bool empty () const`
Test if the queue is empty.
- `const P & front () const`
Give the front site p of the queue; p is the least recently inserted site.
- `bool has (const util::index &i) const`
Test if index i belongs to this site set.
- `bool has (const psite &p) const`
Test if p belongs to this site set.
- `void insert (const P &p)`
Insert a site p (equivalent as 'push').
- `bool is_valid () const`
This set is always valid so it returns true.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `unsigned nsites () const`
Give the number of sites.
- `const P & operator[] (unsigned i) const`
Return the i -th site.
- `p_queue_fast ()`
Constructor without argument.
- `void pop ()`
Pop (remove) the front site p from the queue; p is the least recently inserted site.
- `const P & pop_front ()`
Pop (remove) the front site p from the queue; p is the least recently inserted site and give the front site p of the queue; p is the least recently inserted site.
- `void purge ()`
Purge the queue to save (free) some memory.
- `void push (const P &p)`
Push a site p in the queue.
- `void reserve (typename p_array< P >::size_type n)`
Reserve n cells.

- `const std::vector< P > & std_vector () const`

Return the corresponding std::vector of sites.

10.289.1 Detailed Description

template<typename P> class mln::p_queue_fast< P >

Queue of sites class (based on [p_array](#)).

This container is efficient; FIXME: explain...

The parameter P shall be a site or pseudo-site type.

Definition at line 72 of file p_queue_fast.hh.

10.289.2 Member Typedef Documentation

10.289.2.1 template<typename P > typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 87 of file p_queue_fast.hh.

10.289.2.2 template<typename P > typedef P mln::p_queue_fast< P >::element

Element associated type.

Definition at line 78 of file p_queue_fast.hh.

10.289.2.3 template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 84 of file p_queue_fast.hh.

10.289.2.4 template<typename P > typedef P mln::p_queue_fast< P >::i_element

Insertion element associated type.

Definition at line 121 of file p_queue_fast.hh.

10.289.2.5 template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter

[Site_Iterator](#) associated type.

Definition at line 90 of file p_queue_fast.hh.

10.289.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_queue_fast< P>::psite`

Psite associated type.

Definition at line 81 of file p_queue_fast.hh.

10.289.3 Constructor & Destructor Documentation

10.289.3.1 `template<typename P> mln::p_queue_fast< P>::p_queue_fast () [inline]`

Constructor without argument.

Definition at line 170 of file p_queue_fast.hh.

10.289.4 Member Function Documentation

10.289.4.1 `template<typename P> void mln::p_queue_fast< P>::clear () [inline]`

Clear the queue.

Definition at line 297 of file p_queue_fast.hh.

10.289.4.2 `template<typename P> bool mln::p_queue_fast< P>::compute_has (const P & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 222 of file p_queue_fast.hh.

10.289.4.3 `template<typename P> bool mln::p_queue_fast< P>::empty () const [inline]`

Test if the queue is empty.

Definition at line 250 of file p_queue_fast.hh.

10.289.4.4 `template<typename P> const P & mln::p_queue_fast< P>::front () const [inline]`

Give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 277 of file p_queue_fast.hh.

Referenced by `mln::p_queue_fast< P>::pop_front()`.

10.289.4.5 `template<typename P> bool mln::p_queue_fast< P>::has (const psite & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 201 of file p_queue_fast.hh.

References `mln::p_queue_fast< P>::nsites()`.

10.289.4.6 `template<typename P> bool mln::p_queue_fast< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this site set.

Definition at line 214 of file p_queue_fast.hh.

References mln::p_queue_fast< P >::nsites().

10.289.4.7 `template<typename P> void mln::p_queue_fast< P >::insert (const P & p) [inline]`

Insert a site *p* (equivalent as 'push').

Definition at line 314 of file p_queue_fast.hh.

References mln::p_queue_fast< P >::push().

10.289.4.8 `template<typename P> bool mln::p_queue_fast< P >::is_valid () const [inline]`

This set is always valid so it returns true.

Definition at line 233 of file p_queue_fast.hh.

10.289.4.9 `template<typename P> std::size_t mln::p_queue_fast< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 330 of file p_queue_fast.hh.

10.289.4.10 `template<typename P> unsigned mln::p_queue_fast< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 241 of file p_queue_fast.hh.

Referenced by mln::p_queue_fast< P >::has(), and mln::p_queue_fast< P >::operator[]().

10.289.4.11 `template<typename P> const P & mln::p_queue_fast< P >::operator[] (unsigned i) const [inline]`

Return the *i*-th site.

Definition at line 305 of file p_queue_fast.hh.

References mln::p_queue_fast< P >::nsites().

10.289.4.12 `template<typename P> void mln::p_queue_fast< P >::pop () [inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.

Definition at line 268 of file p_queue_fast.hh.

Referenced by `mln::p_queue_fast< P >::pop_front()`.

10.289.4.13 `template<typename P> const P & mln::p_queue_fast< P >::pop_front ()` [inline]

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 286 of file `p_queue_fast.hh`.

References `mln::p_queue_fast< P >::front()`, and `mln::p_queue_fast< P >::pop()`.

10.289.4.14 `template<typename P> void mln::p_queue_fast< P >::purge ()` [inline]

Purge the queue to save (free) some memory.

Definition at line 187 of file `p_queue_fast.hh`.

10.289.4.15 `template<typename P> void mln::p_queue_fast< P >::push (const P & p)` [inline]

Push a site `p` in the queue.

Definition at line 259 of file `p_queue_fast.hh`.

Referenced by `mln::p_queue_fast< P >::insert()`.

10.289.4.16 `template<typename P> void mln::p_queue_fast< P >::reserve (typename p_array< P >::size_type n)` [inline]

Reserve `n` cells.

Definition at line 179 of file `p_queue_fast.hh`.

10.289.4.17 `template<typename P> const std::vector< P > & mln::p_queue_fast< P >::std_vector () const` [inline]

Return the corresponding `std::vector` of sites.

Definition at line 322 of file `p_queue_fast.hh`.

10.290 `mln::p_run< P >` Class Template Reference

[Point](#) set class in run.

```
#include <p_run.hh>
```

Inherits `site_set_base_< P, p_run< P > >`.

Public Types

- `typedef p_run_bkd_piter_< P > bkd_piter`

Backward [Site_Iterator](#) associated type.

- typedef P [element](#)
Element associated type.
- typedef p_run_fwd_piter_< P > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef p_run_psite< P > [psite](#)
Psite associated type.
- typedef [mln::box](#)< P > [q_box](#)
[Box](#) associated type.

Public Member Functions

- [mln::box](#)< P > [bbox](#) () const
Give the exact bounding box.
- P [end](#) () const
Return (compute) the ending point.
- bool [has](#) (const P &p) const
*Test if *p* belongs to this point set.*
- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this point set.*
- bool [has_index](#) (unsigned short i) const
*Test if index *i* belongs to this point set.*
- void [init](#) (const P &start, unsigned short len)
Set the starting point.
- bool [is_valid](#) () const
Test if this run is valid, i.e., with length > 0.
- unsigned short [length](#) () const
Give the length of the run.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.

- `P operator[]` (unsigned short i) const
Return the i -th point.
- `p_run ()`
Constructor without argument.
- `p_run` (const P &start, unsigned short len)
Constructor.
- `p_run` (const P &start, const P &end)
Constructor.
- const P & `start ()` const
Return the starting point.

10.290.1 Detailed Description

`template<typename P> class mln::p_run< P >`

[Point](#) set class in run. This is a mathematical set of points (not a multi-set). The parameter `P` shall be a [Point](#) type.

Definition at line 86 of file `p_run.hh`.

10.290.2 Member Typedef Documentation

10.290.2.1 `template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 101 of file `p_run.hh`.

10.290.2.2 `template<typename P> typedef P mln::p_run< P >::element`

Element associated type.

Definition at line 91 of file `p_run.hh`.

10.290.2.3 `template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 98 of file `p_run.hh`.

10.290.2.4 `template<typename P> typedef fwd_piter mln::p_run< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 104 of file `p_run.hh`.

10.290.2.5 template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite

Psite associated type.

Definition at line 95 of file p_run.hh.

10.290.2.6 template<typename P> typedef mln::box<P> mln::p_run< P >::q_box

Box associated type.

Definition at line 149 of file p_run.hh.

10.290.3 Constructor & Destructor Documentation**10.290.3.1 template<typename P> mln::p_run< P >::p_run () [inline]**

Constructor without argument.

Definition at line 223 of file p_run.hh.

10.290.3.2 template<typename P> mln::p_run< P >::p_run (const P & start, unsigned short len) [inline]

Constructor.

Definition at line 230 of file p_run.hh.

References mln::p_run< P >::init().

10.290.3.3 template<typename P> mln::p_run< P >::p_run (const P & start, const P & end) [inline]

Constructor.

Definition at line 238 of file p_run.hh.

10.290.4 Member Function Documentation**10.290.4.1 template<typename P> mln::box< P > mln::p_run< P >::bbox () const [inline]**

Give the exact bounding box.

Definition at line 267 of file p_run.hh.

References mln::p_run< P >::end().

10.290.4.2 template<typename P> P mln::p_run< P >::end () const [inline]

Return (compute) the ending point.

Definition at line 348 of file p_run.hh.

References mln::point< G, C >::last_coord().

Referenced by `mln::p_run< P >::bbox()`.

10.290.4.3 `template<typename P> bool mln::p_run< P >::has (const psite & p) const [inline]`

Test if `p` belongs to this point set.

Definition at line 276 of file `p_run.hh`.

10.290.4.4 `template<typename P> bool mln::p_run< P >::has (const P & p) const [inline]`

Test if `p` belongs to this point set.

Definition at line 289 of file `p_run.hh`.

References `mln::p_run< P >::is_valid()`.

10.290.4.5 `template<typename P> bool mln::p_run< P >::has_index (unsigned short i) const [inline]`

Test if index `i` belongs to this point set.

Definition at line 302 of file `p_run.hh`.

10.290.4.6 `template<typename P> void mln::p_run< P >::init (const P & start, unsigned short len) [inline]`

Set the starting point.

Definition at line 249 of file `p_run.hh`.

Referenced by `mln::p_run< P >::p_run()`.

10.290.4.7 `template<typename P> bool mln::p_run< P >::is_valid () const [inline]`

Test if this run is valid, i.e., with `length > 0`.

Definition at line 259 of file `p_run.hh`.

Referenced by `mln::p_run< P >::has()`, `mln::p_run< P >::length()`, `mln::p_run< P >::nsites()`, and `mln::p_run< P >::operator[]()`.

10.290.4.8 `template<typename P> unsigned short mln::p_run< P >::length () const [inline]`

Give the length of the run.

Definition at line 319 of file `p_run.hh`.

References `mln::p_run< P >::is_valid()`.

10.290.4.9 `template<typename P> std::size_t mln::p_run< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 358 of file p_run.hh.

10.290.4.10 `template<typename P> unsigned mln::p_run< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 310 of file p_run.hh.

References mln::p_run< P >::is_valid().

10.290.4.11 `template<typename P> P mln::p_run< P >::operator[] (unsigned short i) const [inline]`

Return the *i*-th point.

Definition at line 328 of file p_run.hh.

References mln::p_run< P >::is_valid(), and mln::point< G, C >::last_coord().

10.290.4.12 `template<typename P> const P & mln::p_run< P >::start () const [inline]`

Return the starting point.

Definition at line 340 of file p_run.hh.

10.291 mln::p_set< P > Class Template Reference

Mathematical set of sites (based on [util::set](#)).

```
#include <p_set.hh>
```

Inherits `site_set_base_< P, p_set< P > >`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `P i_element`
Insertion element associated type.

- typedef `fwd_piter` `piter`
Site_iterator associated type.
- typedef `p_indexed_psite`< `self_` > `psite`
Psite associated type.
- typedef `P_r_element`
Removal element associated type.

Public Member Functions

- void `clear` ()
Clear this set.
- bool `has` (const `psite` &p) const
Test if psite p belongs to this point set.
- bool `has` (const util::index &i) const
Test if index i belongs to this point set.
- bool `has` (const P &p) const
Test if p belongs to this point set.
- void `insert` (const P &p)
Insert a site p.
- bool `is_valid` () const
Test this set validity so returns always true.
- std::size_t `memory_size` () const
Return the size of this site set in memory.
- unsigned `nsites` () const
Give the number of sites.
- const P & `operator[]` (unsigned i) const
Return the i-th site.
- `p_set` ()
Constructor.
- void `remove` (const P &p)
Remove a site p.
- const std::vector< P > & `std_vector` () const
Return the corresponding std::vector of sites.
- const util::set< P > & `util_set` () const
Return the corresponding util::set of sites.

10.291.1 Detailed Description

template<typename P> class mln::p_set< P >

Mathematical set of sites (based on [util::set](#)). This is a mathematical set of sites (not a multi-set).

The parameter P shall be a site or pseudo-site type.

Definition at line 70 of file p_set.hh.

10.291.2 Member Typedef Documentation

10.291.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 85 of file p_set.hh.

10.291.2.2 template<typename P> typedef P mln::p_set< P >::element

Element associated type.

Definition at line 76 of file p_set.hh.

10.291.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 82 of file p_set.hh.

10.291.2.4 template<typename P> typedef P mln::p_set< P >::i_element

Insertion element associated type.

Definition at line 113 of file p_set.hh.

10.291.2.5 template<typename P> typedef fwd_piter mln::p_set< P >::piter

[Site_Iterator](#) associated type.

Definition at line 88 of file p_set.hh.

10.291.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite

Psite associated type.

Definition at line 79 of file p_set.hh.

10.291.2.7 template<typename P> typedef P mln::p_set< P >::r_element

Removal element associated type.

Definition at line 119 of file p_set.hh.

10.291.3 Constructor & Destructor Documentation

10.291.3.1 `template<typename P> mln::p_set<P>::p_set () [inline]`

Constructor.

Definition at line 152 of file p_set.hh.

10.291.4 Member Function Documentation

10.291.4.1 `template<typename P> void mln::p_set<P>::clear () [inline]`

Clear this set.

Definition at line 219 of file p_set.hh.

10.291.4.2 `template<typename P> bool mln::p_set<P>::has (const psite & p) const [inline]`

Test if psite *p* belongs to this point set.

Definition at line 167 of file p_set.hh.

10.291.4.3 `template<typename P> bool mln::p_set<P>::has (const P & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 159 of file p_set.hh.

10.291.4.4 `template<typename P> bool mln::p_set<P>::has (const util::index & i) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 179 of file p_set.hh.

References `mln::p_set<P>::nsites()`.

10.291.4.5 `template<typename P> void mln::p_set<P>::insert (const P & p) [inline]`

Insert a site *p*.

Definition at line 203 of file p_set.hh.

Referenced by `mln::convert::to_p_set()`.

10.291.4.6 `template<typename P> bool mln::p_set<P>::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 187 of file p_set.hh.

10.291.4.7 template<typename P > std::size_t mln::p_set< P >::memory_size () const [inline]

Return the size of this site set in memory.

Definition at line 236 of file p_set.hh.

10.291.4.8 template<typename P > unsigned mln::p_set< P >::nsites () const [inline]

Give the number of sites.

Definition at line 195 of file p_set.hh.

Referenced by mln::p_key< K, P >::change_key(), mln::p_set< P >::has(), mln::p_set< P >::operator[](), and mln::p_key< K, P >::remove_key().

10.291.4.9 template<typename P > const P & mln::p_set< P >::operator[] (unsigned i) const [inline]

Return the *i*-th site.

Definition at line 227 of file p_set.hh.

References mln::p_set< P >::nsites().

10.291.4.10 template<typename P > void mln::p_set< P >::remove (const P & p) [inline]

Remove a site *p*.

Definition at line 211 of file p_set.hh.

10.291.4.11 template<typename P > const std::vector< P > & mln::p_set< P >::std_vector () const [inline]

Return the corresponding std::vector of sites.

Definition at line 244 of file p_set.hh.

10.291.4.12 template<typename P > const util::set< P > & mln::p_set< P >::util_set () const [inline]

Return the corresponding [util::set](#) of sites.

Definition at line 252 of file p_set.hh.

10.292 mln::p_transformed< S, F > Class Template Reference

[Site](#) set transformed through a function.

```
#include <p_transformed.hh>
```

Inherits `site_set_base_< S::psite, p_transformed< S, F > >`.

Public Types

- typedef [p_transformed_piter](#)< typename S::bkd_piter, S, F > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef S::element [element](#)
Element associated type.
- typedef [p_transformed_piter](#)< typename S::fwd_piter, S, F > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef S::psite [psite](#)
Psite associated type.

Public Member Functions

- const F & [function](#) () const
Return the transformation function.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the subset.
- bool [is_valid](#) () const
Test if this site set is valid.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- [p_transformed](#) ()
Constructor without argument.
- [p_transformed](#) (const S &s, const F &f)
Constructor with a site set s and a predicate f .
- const S & [primary_set](#) () const
Return the primary set.

10.292.1 Detailed Description

template<typename S, typename F> class mln::p_transformed< S, F >

[Site](#) set transformed through a function. Parameter S is a site set type; parameter F is a function from site to site.

Definition at line 82 of file [p_transformed.hh](#).

10.292.2 Member Typedef Documentation

10.292.2.1 `template<typename S, typename F> typedef p_transformed_piter<typename S
::bkd_piter, S, F> mln::p_transformed< S, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 101 of file p_transformed.hh.

10.292.2.2 `template<typename S, typename F> typedef S ::element mln::p_transformed< S, F
>::element`

Element associated type.

Definition at line 91 of file p_transformed.hh.

10.292.2.3 `template<typename S, typename F> typedef p_transformed_piter<typename S
::fwd_piter, S, F> mln::p_transformed< S, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 98 of file p_transformed.hh.

10.292.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F
>::piter`

[Site_Iterator](#) associated type.

Definition at line 104 of file p_transformed.hh.

10.292.2.5 `template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F
>::psite`

Psite associated type.

Definition at line 95 of file p_transformed.hh.

10.292.3 Constructor & Destructor Documentation

10.292.3.1 `template<typename S , typename F > mln::p_transformed< S, F >::p_transformed (
const S & s, const F & f) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 163 of file p_transformed.hh.

10.292.3.2 `template<typename S , typename F > mln::p_transformed< S, F >::p_transformed (
) [inline]`

Constructor without argument.

Definition at line 157 of file p_transformed.hh.

10.292.4 Member Function Documentation

10.292.4.1 `template<typename S , typename F > const F & mln::p_transformed< S, F >::function () const [inline]`

Return the transformation function.

Definition at line 206 of file p_transformed.hh.

10.292.4.2 `template<typename S , typename F > bool mln::p_transformed< S, F >::has (const psite & p) const [inline]`

Test if p belongs to the subset.

Definition at line 172 of file p_transformed.hh.

10.292.4.3 `template<typename S , typename F > bool mln::p_transformed< S, F >::is_valid () const [inline]`

Test if this site set is valid.

Definition at line 182 of file p_transformed.hh.

10.292.4.4 `template<typename S , typename F > std::size_t mln::p_transformed< S, F >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 190 of file p_transformed.hh.

10.292.4.5 `template<typename S , typename F > const S & mln::p_transformed< S, F >::primary_set () const [inline]`

Return the primary set.

Definition at line 198 of file p_transformed.hh.

Referenced by `mln::p_transformed_piter< Pi, S, F >::change_target()`.

10.293 mln::p_transformed_piter< Pi, S, F > Struct Template Reference

[Iterator](#) on p_transformed<S,F>.

```
#include <p_transformed_piter.hh>
```

Inherits `mln::internal::site_set_iterator_base< p_transformed< S, F >,p_transformed_piter< Pi, S, F > >`.

Public Member Functions

- void [change_target](#) (const [p_transformed](#)< S, F > &s)

Change the set site targeted by this iterator.

- void [next](#) ()
Go to the next element.
- [p_transformed_piter](#) (const [p_transformed](#)< S, F > &s)
Constructor from a site set.
- [p_transformed_piter](#) ()
Constructor without argument.

10.293.1 Detailed Description

template<typename Pi, typename S, typename F> struct mln::p_transformed_piter< Pi, S, F >

[Iterator](#) on [p_transformed](#)<S,F>. Parameter S is a site set type; parameter F is a function from point to Boolean.

See also

[mln::p_transformed](#)

Definition at line 50 of file [p_transformed_piter.hh](#).

10.293.2 Constructor & Destructor Documentation

10.293.2.1 template<typename Pi, typename S, typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter () [inline]

Constructor without argument.

Definition at line 93 of file [p_transformed_piter.hh](#).

10.293.2.2 template<typename Pi, typename S, typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter (const [p_transformed](#)< S, F > & s) [inline]

Constructor from a site set.

Definition at line 99 of file [p_transformed_piter.hh](#).

References [mln::p_transformed_piter< Pi, S, F >::change_target\(\)](#).

10.293.3 Member Function Documentation

10.293.3.1 template<typename Pi, typename S, typename F > void mln::p_transformed_piter< Pi, S, F >::change_target (const [p_transformed](#)< S, F > & s) [inline]

Change the set site targeted by this iterator.

Definition at line 143 of file [p_transformed_piter.hh](#).

References [mln::p_transformed< S, F >::primary_set\(\)](#).

Referenced by `mln::p_transformed_piter< Pi, S, F >::p_transformed_piter()`.

10.293.3.2 `template<typename E > void mln::Site_Iterator< E >::next () [inline, inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

Definition at line 92 of file `site_iterator.hh`.

10.294 `mln::p_vaccess< V, S >` Class Template Reference

[Site](#) set in which sites are grouped by their associated value.

```
#include <p_vaccess.hh>
```

Inherits `site_set_base_< S::site, p_vaccess< V, S > >`, and `site_set_impl< S >`.

Public Types

- `typedef p_double_piter< self_, typename vset::bkd_viter, typename S::bkd_piter > bkd_piter`
Backward [Site_Iterator](#) associated type.
- `typedef S::element element`
Element associated type.
- `typedef p_double_piter< self_, typename vset::fwd_viter, typename S::fwd_piter > fwd_piter`
Forward [Site_Iterator](#) associated type.
- `typedef std::pair< V, element > i_element`
Insertion element associated type.
- `typedef fwd_piter piter`
[Site_Iterator](#) associated type.
- `typedef S pset`
Inner site set associated type.
- `typedef p_double_psite< self_, S > psite`
Psite associated type.
- `typedef V value`

Value associated type.

- typedef `mln::value::set< V > vset`
Value_Set associated type.

Public Member Functions

- bool `has` (const `psite` &p) const
Test if p belongs to this site set.
- bool `has` (const V &v, const typename S::psite &p) const
Test if the couple (value v, psite p) belongs to this site set.
- void `insert` (const `i_element` &v_e)
Insert a pair v_e (value v, element e).
- void `insert` (const V &v, const `element` &e)
Insert e at value v.
- bool `is_valid` () const
Test if this site set is valid.
- std::size_t `memory_size` () const
Return the size of this site set in memory.
- const S & `operator()` (const V &v) const
Return the site set at value v.
- `p_vaccess` ()
Constructor.
- const `mln::value::set< V > & values` () const
Give the set of values.

10.294.1 Detailed Description

`template<typename V, typename S> class mln::p_vaccess< V, S >`

[Site](#) set in which sites are grouped by their associated value.

Definition at line 70 of file `p_vaccess.hh`.

10.294.2 Member Typedef Documentation

10.294.2.1 `template<typename V , typename S > typedef p_double_piter<self_, typename vset
::bkd_viter, typename S ::bkd_piter> mln::p_vaccess< V, S >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 94 of file p_vaccess.hh.

10.294.2.2 `template<typename V , typename S > typedef S ::element mln::p_vaccess< V, S >::element`

Element associated type.

Definition at line 117 of file p_vaccess.hh.

10.294.2.3 `template<typename V , typename S > typedef p_double_piter<self_, typename vset ::fwd_viter, typename S ::fwd_piter> mln::p_vaccess< V, S >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 91 of file p_vaccess.hh.

10.294.2.4 `template<typename V , typename S > typedef std::pair<V, element> mln::p_vaccess< V, S >::i_element`

Insertion element associated type.

Definition at line 120 of file p_vaccess.hh.

10.294.2.5 `template<typename V , typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter`

[Site_Iterator](#) associated type.

Definition at line 97 of file p_vaccess.hh.

10.294.2.6 `template<typename V , typename S > typedef S mln::p_vaccess< V, S >::pset`

Inner site set associated type.

Definition at line 85 of file p_vaccess.hh.

10.294.2.7 `template<typename V , typename S > typedef p_double_psite<self_, S> mln::p_vaccess< V, S >::psite`

Psite associated type.

Definition at line 88 of file p_vaccess.hh.

10.294.2.8 `template<typename V , typename S > typedef V mln::p_vaccess< V, S >::value`

[Value](#) associated type.

Definition at line 78 of file p_vaccess.hh.

10.294.2.9 `template<typename V , typename S > typedef mln::value::set<V> mln::p_vaccess< V, S >::vset`

Value_Set associated type.

Definition at line 81 of file p_vaccess.hh.

10.294.3 Constructor & Destructor Documentation

10.294.3.1 `template<typename V , typename S > mln::p_vaccess< V, S >::p_vaccess ()
[inline]`

Constructor.

Definition at line 163 of file p_vaccess.hh.

10.294.4 Member Function Documentation

10.294.4.1 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has (const V &
v, const typename S::psite & p) const [inline]`

Test if the couple (value v, psite p) belongs to this site set.

Definition at line 189 of file p_vaccess.hh.

10.294.4.2 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has (const psite
& p) const [inline]`

Test if p belongs to this site set.

Definition at line 180 of file p_vaccess.hh.

10.294.4.3 `template<typename V , typename S > void mln::p_vaccess< V, S >::insert (const
i_element & v_e) [inline]`

Insert a pair v_e (value v, element e).

Definition at line 216 of file p_vaccess.hh.

10.294.4.4 `template<typename V , typename S > void mln::p_vaccess< V, S >::insert (const V
& v, const element & e) [inline]`

Insert e at value v.

Definition at line 206 of file p_vaccess.hh.

10.294.4.5 `template<typename V , typename S > bool mln::p_vaccess< V, S >::is_valid () const
[inline]`

Test if this site set is valid.

Definition at line 197 of file p_vaccess.hh.

10.294.4.6 `template<typename V , typename S > std::size_t mln::p_vaccess< V, S
>::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 242 of file p_vaccess.hh.

10.294.4.7 `template<typename V , typename S > const S & mln::p_vaccess< V, S >::operator() (const V & v) const [inline]`

Return the site set at value v.

Definition at line 234 of file p_vaccess.hh.

10.294.4.8 `template<typename V , typename S > const mln::value::set< V > & mln::p_vaccess< V, S >::values () const [inline]`

Give the set of values.

Definition at line 254 of file p_vaccess.hh.

10.295 mln::p_vertices< G, F > Class Template Reference

[Site](#) set based mapping graph vertices to sites.

```
#include <p_vertices.hh>
```

Inherits `site_set_base_< F::result, p_vertices< G, F > >`.

Public Types

- typedef F [fun_t](#)
Function associated type.
- typedef [util::vertex](#)< G > [graph_element](#)
Type of graph element this site set focuses on.
- typedef G [graph_t](#)
Graph associated type.
- typedef [util::vertex](#)< G > [vertex](#)
Type of graph vertex.
- typedef `super_::site` [element](#)
Associated types.
- typedef `p_vertices_psite`< G, F > [psite](#)
Point_Site associated type.
- typedef `p_graph_piter`< [self_](#), `mln_vertex_fwd_iter`(G) > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef `p_graph_piter`< [self_](#), `mln_vertex_bkd_iter`(G) > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.

- typedef fwd_piter piter
Site_Iterator associated type.

Public Member Functions

- bool has (const psite &p) const
Does this site set has p?
- template<typename G2 >
bool has (const util::vertex< G2 > &v) const
Does this site set has v?
- void invalidate ()
Invalidate this site set.
- bool is_valid () const
Test this site set validity.
- std::size_t memory_size () const
*Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>);
bool has(unsigned vertex_id) const;.*
- unsigned nsites () const
Return The number of points (sites) of the set, i.e., the number of vertices.
- unsigned nvertices () const
Return The number of vertices in the graph.
- p_vertices (const Graph< G > &gr)
Construct a graph psite set from a graph of points.
- p_vertices (const Graph< G > &gr, const Function< F > &f)
Construct a graph psite set from a graph of points.
- template<typename F2 >
p_vertices (const p_vertices< G, F2 > &other)
Copy constructor.
- p_vertices ()
Constructor without argument.
- template<typename F2 >
p_vertices (const Graph< G > &gr, const Function< F2 > &f)
Construct a graph psite set from a graph of points.
- F::result operator() (const psite &p) const
Return the value associated to an element of this site set.

- `const G & graph () const`
Accessors.
- `const F & function () const`
Return the association function.

10.295.1 Detailed Description

`template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> class mln::p_vertices< G, F >`

[Site](#) set based mapping graph vertices to sites.

Definition at line 71 of file p_vertices.hh.

10.295.2 Member Typedef Documentation

10.295.2.1 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef p_graph_piter< self_, mln_vertex_bkd_iter(G) > mln::p_vertices< G, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 132 of file p_vertices.hh.

10.295.2.2 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef super_::site mln::p_vertices< G, F >::element`

Associated types.

Element associated type.

Definition at line 123 of file p_vertices.hh.

10.295.2.3 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef F mln::p_vertices< G, F >::fun_t`

[Function](#) associated type.

Definition at line 84 of file p_vertices.hh.

10.295.2.4 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef p_graph_piter< self_, mln_vertex_fwd_iter(G) > mln::p_vertices< G, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 129 of file p_vertices.hh.

10.295.2.5 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 91 of file p_vertices.hh.

10.295.2.6 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef G mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 81 of file p_vertices.hh.

10.295.2.7 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef fwd_piter mln::p_vertices< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 135 of file p_vertices.hh.

10.295.2.8 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_vertices_psite<G,F> mln::p_vertices< G, F >::psite`

[Point_Site](#) associated type.

Definition at line 126 of file p_vertices.hh.

10.295.2.9 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::vertex`

Type of graph vertex.

Definition at line 87 of file p_vertices.hh.

10.295.3 Constructor & Destructor Documentation

10.295.3.1 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ()
[inline]`

Constructor without argument.

Definition at line 220 of file p_vertices.hh.

10.295.3.2 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices (const
Graph< G > & gr) [inline]`

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built. The identity function is used.

Definition at line 226 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.295.3.3 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices (const Graph< G > & gr, const Function< F > & f) [inline]`

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built.

f the function which maps a vertex to a site.

Definition at line 238 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.295.3.4 `template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices (const Graph< G > & gr, const Function< F2 > & f) [inline]`

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built.

f the function which maps a vertex to a site. It must be convertible to the function type F.

Definition at line 248 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.295.3.5 `template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices (const p_vertices< G, F2 > & other) [inline]`

Copy constructor.

Definition at line 260 of file p_vertices.hh.

References mln::p_vertices< G, F >::function(), mln::p_vertices< G, F >::graph(), and mln::p_vertices< G, F >::is_valid().

10.295.4 Member Function Documentation

10.295.4.1 `template<typename G , typename F > const F & mln::p_vertices< G, F >::function () const [inline]`

Return the association function.

Definition at line 385 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::p_vertices().

10.295.4.2 `template<typename G , typename F > const G & mln::p_vertices< G, F >::graph ()
const [inline]`

Accessors.

Return the graph associated to this site set (const version)

Definition at line 376 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

Referenced by mln::debug::draw_graph(), mln::operator==(), and mln::p_vertices< G, F >::p_vertices().

10.295.4.3 `template<typename G , typename F > bool mln::p_vertices< G, F >::has (const psite
& p) const [inline]`

Does this site set has *p*?

Definition at line 304 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.295.4.4 `template<typename G , typename F > template<typename G2 > bool
mln::p_vertices< G, F >::has (const util::vertex< G2 > & v) const [inline]`

Does this site set has *v*?

Definition at line 314 of file p_vertices.hh.

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is_valid(), and mln::p_vertices< G, F >::is_valid().

10.295.4.5 `template<typename G , typename F > void mln::p_vertices< G, F >::invalidate ()
[inline]`

Invalidate this site set.

Definition at line 296 of file p_vertices.hh.

10.295.4.6 `template<typename G , typename F > bool mln::p_vertices< G, F >::is_valid ()
const [inline]`

Test this site set validity.

Definition at line 288 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::graph(), mln::p_vertices< G, F >::has(), and mln::p_vertices< G, F >::p_vertices().

10.295.4.7 `template<typename G , typename F > std::size_t mln::p_vertices< G, F
>::memory_size () const [inline]`

Does this site set has *vertex_id*? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.

Definition at line 339 of file p_vertices.hh.

10.295.4.8 `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nsites ()
const [inline]`

Return The number of points (sites) of the set, i.e., the number of *vertices*.

Required by the `mln::Point_Set` concept.

Definition at line 272 of file `p_vertices.hh`.

References `mln::p_vertices< G, F >::nvertices()`.

10.295.4.9 `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nvertices () const [inline]`

Return The number of vertices in the graph.

Definition at line 280 of file `p_vertices.hh`.

Referenced by `mln::p_vertices< G, F >::nsites()`.

10.295.4.10 `template<typename G , typename F > F::result mln::p_vertices< G, F >::operator()
(const psite & p) const [inline]`

Return the value associated to an element of this site set.

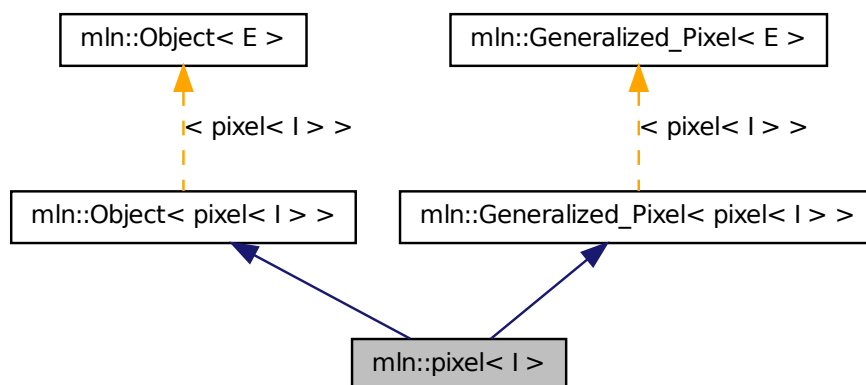
Definition at line 349 of file `p_vertices.hh`.

10.296 mln::pixel< I > Struct Template Reference

Generic pixel class.

```
#include <pixel.hh>
```

Inheritance diagram for `mln::pixel< I >`:



Public Member Functions

- void [change_to](#) (const typename I::psite &p)
Change the pixel to the one at point p.
- bool [is_valid](#) () const
Test if this pixel is valid.
- [pixel](#) (I &image)
Constructor.
- [pixel](#) (I &image, const typename I::psite &p)
Constructor.

10.296.1 Detailed Description

`template<typename I> struct mln::pixel< I >`

Generic pixel class. The parameter is \mathbb{I} the type of the image it belongs to.

Definition at line 50 of file core/pixel.hh.

10.296.2 Constructor & Destructor Documentation

10.296.2.1 `template<typename I> mln::pixel< I >::pixel (I & image) [inline]`

Constructor.

Definition at line 75 of file core/pixel.hh.

10.296.2.2 `template<typename I> mln::pixel< I >::pixel (I & image, const typename I::psite & p) [inline]`

Constructor.

Definition at line 82 of file core/pixel.hh.

References `mln::pixel< I >::change_to()`.

10.296.3 Member Function Documentation

10.296.3.1 `template<typename I> void mln::pixel< I >::change_to (const typename I::psite & p) [inline]`

Change the pixel to the one at point p.

Definition at line 92 of file core/pixel.hh.

Referenced by `mln::pixel< I >::pixel()`.

10.296.3.2 `template<typename I> bool mln::pixel< I>::is_valid () const [inline]`

Test if this pixel is valid.

Definition at line 101 of file `core/pixel.hh`.

10.297 `mln::plain< I>` Class Template Reference

Prevents an image from sharing its data.

```
#include <plain.hh>
```

Inherits `image_identity< I, I::domain_t, plain< I>>`.

Public Types

- typedef `plain< tag::image_< I>>` `skeleton`
Skeleton.

Public Member Functions

- `operator I () const`
Conversion into an image with type `I`.
- `plain< I> & operator= (const I &ima)`
Assignment operator from an image `ima`.
- `plain< I> & operator= (const plain< I> &rhs)`
Assignment operator.
- `plain (const plain< I> &rhs)`
Copy constructor.
- `plain ()`
Constructor without argument.
- `plain (const I &ima)`
Copy constructor from an image `ima`.

10.297.1 Detailed Description

```
template<typename I> class mln::plain< I>
```

Prevents an image from sharing its data. While assigned to another image, its data is duplicated.

Definition at line 82 of file `plain.hh`.

10.297.2 Member Typedef Documentation

10.297.2.1 `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

Definition at line 93 of file plain.hh.

10.297.3 Constructor & Destructor Documentation

10.297.3.1 `template<typename I> mln::plain< I >::plain () [inline]`

Constructor without argument.

Definition at line 141 of file plain.hh.

10.297.3.2 `template<typename I> mln::plain< I >::plain (const plain< I > & rhs) [inline]`

Copy constructor.

Definition at line 147 of file plain.hh.

10.297.3.3 `template<typename I> mln::plain< I >::plain (const I & ima) [inline]`

Copy constructor from an image *ima*.

Definition at line 156 of file plain.hh.

10.297.4 Member Function Documentation

10.297.4.1 `template<typename I> mln::plain< I >::operator I () const [inline]`

Conversion into an image with type *I*.

Definition at line 197 of file plain.hh.

References `mln::duplicate()`.

10.297.4.2 `template<typename I> plain< I > & mln::plain< I >::operator= (const plain< I > & rhs) [inline]`

Assignment operator.

Definition at line 174 of file plain.hh.

10.297.4.3 `template<typename I> plain< I > & mln::plain< I >::operator= (const I & ima) [inline]`

Assignment operator from an image *ima*.

Definition at line 187 of file plain.hh.

10.298 mln::Point< P > Struct Template Reference

Base class for implementation of point classes.

```
#include <point.hh>
```

Inherits Point_Site< P >.

Public Types

- typedef P [point](#)

The associated point type is itself.

Public Member Functions

- const P & [to_point](#) () const

It is a [Point](#) so it returns itself.

Related Functions

(Note that these are not member functions.)

- template<typename P , typename D >
P & [operator+=](#) ([Point](#)< P > &p, const [Dpoint](#)< D > &dp)

Shift a point by a delta-point dp.

- template<typename P , typename D >
P & [operator-=](#) ([Point](#)< P > &p, const [Dpoint](#)< D > &dp)

Shift a point by the negate of a delta-point dp.

- template<typename P , typename D >
P & [operator/](#) ([Point](#)< P > &p, const value::Scalar< D > &dp)

Divide a point by a scalar s.

10.298.1 Detailed Description

```
template<typename P> struct mln::Point< P >
```

Base class for implementation of point classes. A point is an element of a space.

For instance, [mln::point2d](#) is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 62 of file concept/point.hh.

10.298.2 Member Typedef Documentation

10.298.2.1 `template<typename P > typedef P mln::Point< P >::point`

The associated point type is itself.

Definition at line 66 of file concept/point.hh.

10.298.3 Member Function Documentation

10.298.3.1 `template<typename P > const P & mln::Point< P >::to_point () const [inline]`

It is a [Point](#) so it returns itself.

Definition at line 130 of file concept/point.hh.

10.298.4 Friends And Related Function Documentation

10.298.4.1 `template<typename P , typename D > P & operator+=(Point< P > & p, const Dpoint< D > & dp) [related]`

Shift a point by a delta-point dp.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point p once translated by dp.

Precondition

The type of dp has to be compatible with the type of p.

Definition at line 137 of file concept/point.hh.

10.298.4.2 `template<typename P , typename D > P & operator-= (Point< P > & p, const Dpoint< D > & dp) [related]`

Shift a point by the negate of a delta-point dp.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point p once translated by - dp.

Precondition

The type of dp has to be compatible with the type of p.

Definition at line 149 of file concept/point.hh.

10.298.4.3 `template<typename P , typename D > P & operator/ (Point< P > & p, const value::Scalar< D > & dp) [related]`

Divide a point by a scalar *s*.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A scalar.

Returns

A reference to the point *p* once divided by *s*.

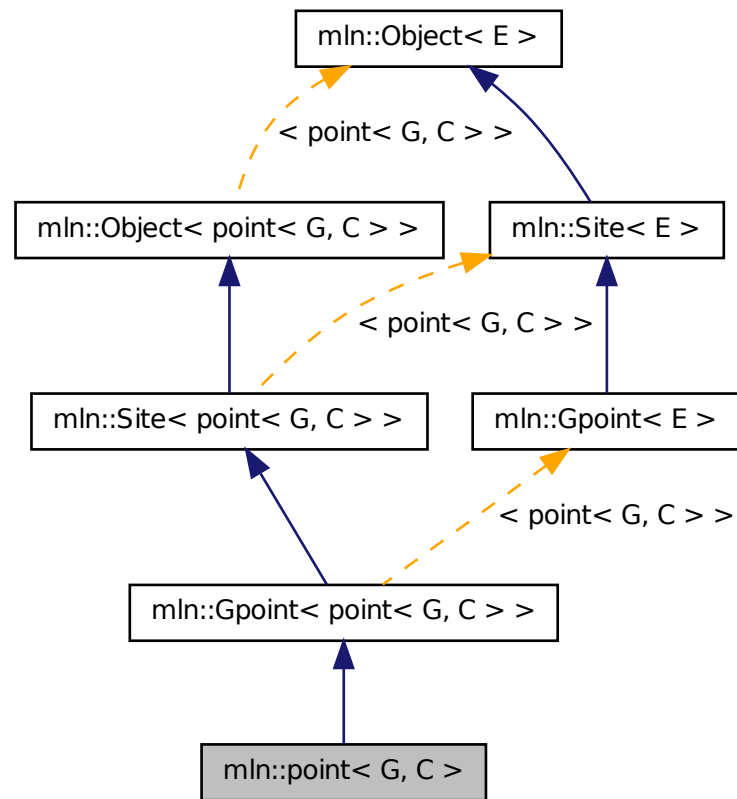
Definition at line 163 of file concept/point.hh.

10.299 mln::point< G, C > Struct Template Reference

Generic point class.

```
#include <point.hh>
```

Inheritance diagram for mln::point< G, C >:



Public Types

- enum { `dim` = `G::dim` }
- typedef `C coord`
Coordinate associated type.
- typedef `dpoint< G, C > delta`
Delta associated type.
- typedef `dpoint< G, C > dpsite`
DPsite associated type.
- typedef `G grid`
Grid associated type.
- typedef `mln::algebra::h_vec< G::dim, float > h_vec`

Algebra hexagonal vector (hvec) associated type.

- `typedef mln::algebra::vec< G::dim, float > vec`
Algebra vector (vec) associated type.

Public Member Functions

- `const C & last_coord () const`
Read-only access to the last coordinate.
- `C & last_coord ()`
Read-write access to the last coordinate.
- `point< G, C > & operator+= (const delta &dp)`
Shifting by dp.
- `point< G, C > & operator-= (const delta &dp)`
Shifting by the inverse of dp.
- `C & operator[] (unsigned i)`
Read-write access to the i-th coordinate value.
- `const C & operator[] (unsigned i) const`
Read-only access to the i-th coordinate value.
- `template<typename F >`
`point (const Function_v2v< F > &f)`
Constructor; coordinates are set by function f.
- `point ()`
Constructor without argument.
- `template<typename C2 >`
`point (const mln::algebra::vec< dim, C2 > &v)`
Constructor from an algebra vector.
- `void set_all (C c)`
Set all coordinates to the value c.
- `h_vec to_h_vec () const`
Transform to point in homogeneous coordinate system.
- `vec to_vec () const`
Explicit conversion towards mln::algebra::vec.
- `point (C ind)`
- `point (const literal::origin_t &)`
Constructors/assignments with literals.

Static Public Member Functions

- static const [point](#)< G, C > & [minus_infty](#) ()
Point with all coordinates set to the minimum value.
- static const [point](#)< G, C > & [plus_infty](#) ()
Point with all coordinates set to the maximum value.

Static Public Attributes

- static const [point](#)< G, C > [origin](#) = all_to(0)
Origin point (all coordinates are 0).

10.299.1 Detailed Description

template<typename G, typename C> struct mln::point< G, C >

Generic point class. Parameters are n the dimension of the space and C the coordinate type in this space.

Definition at line 108 of file point.hh.

10.299.2 Member Typedef Documentation

10.299.2.1 template<typename G, typename C> typedef C mln::point< G, C >::coord

Coordinate associated type.

Definition at line 131 of file point.hh.

10.299.2.2 template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta

Delta associated type.

Definition at line 125 of file point.hh.

10.299.2.3 template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite

DPsite associated type.

Definition at line 128 of file point.hh.

10.299.2.4 template<typename G, typename C> typedef G mln::point< G, C >::grid

Grid associated type.

Definition at line 122 of file point.hh.

10.299.2.5 `template<typename G, typename C> typedef mln::algebra::h_vec<G::dim, float>
mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

Definition at line 137 of file point.hh.

10.299.2.6 `template<typename G, typename C> typedef mln::algebra::vec<G::dim, float>
mln::point< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 134 of file point.hh.

10.299.3 Member Enumeration Documentation

10.299.3.1 `template<typename G, typename C> anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 119 of file point.hh.

10.299.4 Constructor & Destructor Documentation

10.299.4.1 `template<typename G , typename C > mln::point< G, C >::point () [inline]`

Constructor without argument.

Definition at line 420 of file point.hh.

10.299.4.2 `template<typename G , typename C > template<typename C2 > mln::point< G, C
>::point (const mln::algebra::vec< dim, C2 > & v) [inline]`

Constructor from an algebra vector.

Definition at line 427 of file point.hh.

10.299.4.3 `template<typename G , typename C> mln::point< G, C >::point (C ind)
[inline, explicit]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 443 of file point.hh.

10.299.4.4 `template<typename G , typename C> mln::point< G, C >::point (const
literal::origin_t &) [inline]`

Constructors/assignments with literals.

Definition at line 481 of file point.hh.

10.299.4.5 `template<typename G , typename C > template<typename F > mln::point< G, C >::point (const Function_v2v< F > & f) [inline]`

Constructor; coordinates are set by function *f*.

Definition at line 471 of file point.hh.

10.299.5 Member Function Documentation

10.299.5.1 `template<typename G , typename C > const C & mln::point< G, C >::last_coord () const [inline]`

Read-only access to the last coordinate.

Definition at line 402 of file point.hh.

Referenced by `mln::p_run< P >::end()`, `mln::p_run< P >::operator[]()`, and `mln::debug::put_word()`.

10.299.5.2 `template<typename G , typename C > C & mln::point< G, C >::last_coord () [inline]`

Read-write access to the last coordinate.

Definition at line 410 of file point.hh.

10.299.5.3 `template<typename G , typename C > const point< G, C > & mln::point< G, C >::minus_infty () [inline, static]`

[Point](#) with all coordinates set to the minimum value.

Definition at line 627 of file point.hh.

10.299.5.4 `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator+=(const delta & dp) [inline]`

Shifting by *dp*.

Definition at line 544 of file point.hh.

10.299.5.5 `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator-= (const delta & dp) [inline]`

Shifting by the inverse of *dp*.

Definition at line 554 of file point.hh.

10.299.5.6 `template<typename G , typename C > C & mln::point< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 393 of file point.hh.

10.299.5.7 `template<typename G , typename C > const C & mln::point< G, C >::operator[] (unsigned i) const [inline]`

Read-only access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 385 of file point.hh.

10.299.5.8 `template<typename G , typename C > const point< G, C > & mln::point< G, C >::plus_infty () [inline, static]`

[Point](#) with all coordinates set to the maximum value.

Definition at line 618 of file point.hh.

10.299.5.9 `template<typename G , typename C> void mln::point< G, C >::set_all (C c) [inline]`

Set all coordinates to the value *c*.

Definition at line 533 of file point.hh.

10.299.5.10 `template<typename G , typename C > point< G, C >::h_vec mln::point< G, C >::to_h_vec () const [inline]`

Transform to point in homogeneous coordinate system.

Definition at line 592 of file point.hh.

10.299.5.11 `template<typename G , typename C > point< G, C >::vec mln::point< G, C >::to_vec () const [inline]`

Explicit conversion towards `mln::algebra::vec`.

Definition at line 571 of file point.hh.

10.299.6 Member Data Documentation

10.299.6.1 `template<typename G, typename C> const point< G, C > mln::point< G, C >::origin = all_to(0) [static]`

Origin point (all coordinates are 0).

Definition at line 192 of file point.hh.

10.300 mln::Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "proxy".

```
#include <proxy.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Accumulator< E >](#), [mln::internal::graph_iter_base< G, Elt, E >](#), [mln::internal::nbh_iterator_base< G, C, Elt, E >](#), and [mln::Site_Proxy< E >](#).

10.300.1 Detailed Description

`template<typename E> struct mln::Proxy< E >`

Base class for implementation classes of the notion of "proxy".

Definition at line 232 of file core/concept/proxy.hh.

10.301 mln::Proxy< void > Struct Template Reference

[Proxy](#) category flag type.

```
#include <proxy.hh>
```

10.301.1 Detailed Description

`template<> struct mln::Proxy< void >`

[Proxy](#) category flag type.

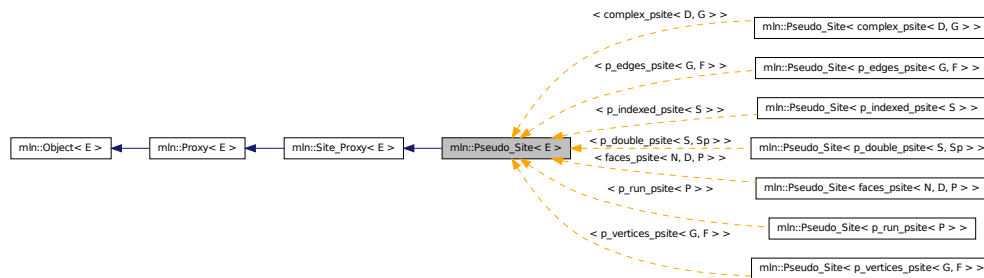
Definition at line 222 of file core/concept/proxy.hh.

10.302 mln::Pseudo_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "pseudo site".

```
#include <pseudo_site.hh>
```

Inheritance diagram for `mln::Pseudo_Site< E >`:



10.302.1 Detailed Description

template<typename E> struct mln::Pseudo_Site< E >

Base class for implementation classes of the notion of "pseudo site". FIXME: Explain...

Definition at line 64 of file `pseudo_site.hh`.

10.303 mln::Pseudo_Site< void > Struct Template Reference

[Pseudo_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

10.303.1 Detailed Description

template<> struct mln::Pseudo_Site< void >

[Pseudo_Site](#) category flag type.

Definition at line 52 of file `pseudo_site.hh`.

10.304 mln::pw::image< F, S > Class Template Reference

A generic point-wise image implementation.

```
#include <image.hh>
```

Inherits `image_base< F, S, image< F, S > >`.

Public Types

- typedef [image](#)< tag::function_< F >, tag::domain_< S > > [skeleton](#)
Skeleton.

Public Member Functions

- [image](#) ()
Constructor without argument.
- [image](#) (const [Function_v2v](#)< F > &f, const [Site_Set](#)< S > &ps)
Constructor.

10.304.1 Detailed Description

template<typename F, typename S> class mln::pw::image< F, S >

A generic point-wise image implementation. Parameter F is a function restricting the domain. Parameter S is the domain type.

Definition at line 92 of file pw/image.hh.

10.304.2 Member Typedef Documentation

10.304.2.1 **template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton**

Skeleton.

Definition at line 99 of file pw/image.hh.

10.304.3 Constructor & Destructor Documentation

10.304.3.1 **template<typename F , typename S > mln::pw::image< F, S >::image ()**
[inline]

Constructor without argument.

Definition at line 169 of file pw/image.hh.

10.304.3.2 **template<typename F , typename S > mln::pw::image< F, S >::image (const Function_v2v< F > & f, const Site_Set< S > & ps)** **[inline]**

Constructor.

Definition at line 175 of file pw/image.hh.

10.305 mln::registration::closest_point_basic< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.305.1 Detailed Description

template<typename P> class mln::registration::closest_point_basic< P >

Closest point functor based on map distance.

Definition at line 240 of file icp.hh.

10.306 mln::registration::closest_point_with_map< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.306.1 Detailed Description

template<typename P> class mln::registration::closest_point_with_map< P >

Closest point functor based on map distance.

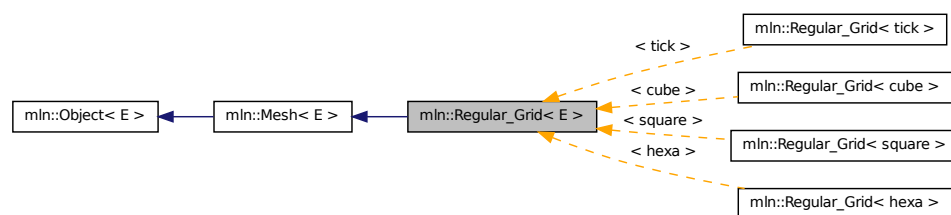
Definition at line 145 of file icp.hh.

10.307 mln::Regular_Grid< E > Struct Template Reference

Base class for implementation classes of regular grids.

```
#include <regular_grid.hh>
```

Inheritance diagram for mln::Regular_Grid< E >:



10.307.1 Detailed Description

template<typename E> struct mln::Regular_Grid< E >

Base class for implementation classes of regular grids.

Definition at line 42 of file regular_grid.hh.

10.308 mln::safe_image< I > Class Template Reference

Makes an image accessible at undefined location.

```
#include <safe.hh>
```

Inherits image_identity< I, I::domain_t, safe_image< I > >.

Public Types

- typedef [safe_image](#)< tag::image_< I > > [skeleton](#)
Skeleton.

Public Member Functions

- [operator safe_image](#)< const I > () const
Const promotion via conversion.

10.308.1 Detailed Description

`template<typename I> class mln::safe_image< I >`

Makes an image accessible at undefined location.

Definition at line 83 of file safe.hh.

10.308.2 Member Typedef Documentation

10.308.2.1 `template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton`

Skeleton.

Definition at line 88 of file safe.hh.

10.308.3 Member Function Documentation

10.308.3.1 `template<typename I> mln::safe_image< I >::operator safe_image< const I > () const [inline]`

Const promotion via conversion.

Definition at line 195 of file safe.hh.

10.309 mln::select::p_of< P > Struct Template Reference

Structure [p_of](#).

```
#include <pix.hh>
```

10.309.1 Detailed Description

template<typename P> struct mln::select::p_of< P >

Structure [p_of](#).

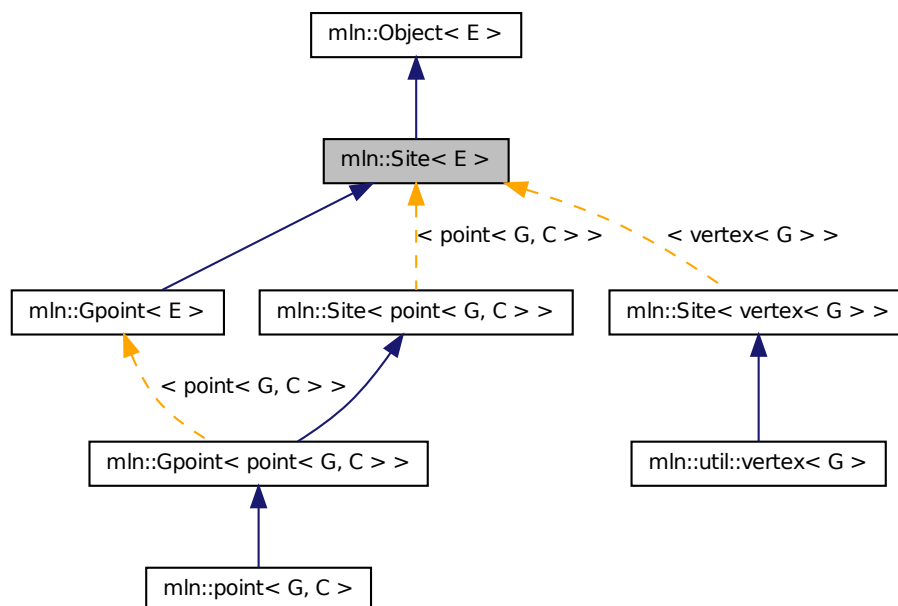
Definition at line 52 of file util/pix.hh.

10.310 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

`#include <site.hh>`

Inheritance diagram for mln::Site< E >:



10.310.1 Detailed Description

template<typename E> struct mln::Site< E >

Base class for classes that are explicitly sites.

Definition at line 55 of file site.hh.

10.311 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

10.311.1 Detailed Description

template<> struct mln::Site< void >

[Site](#) category flag type.

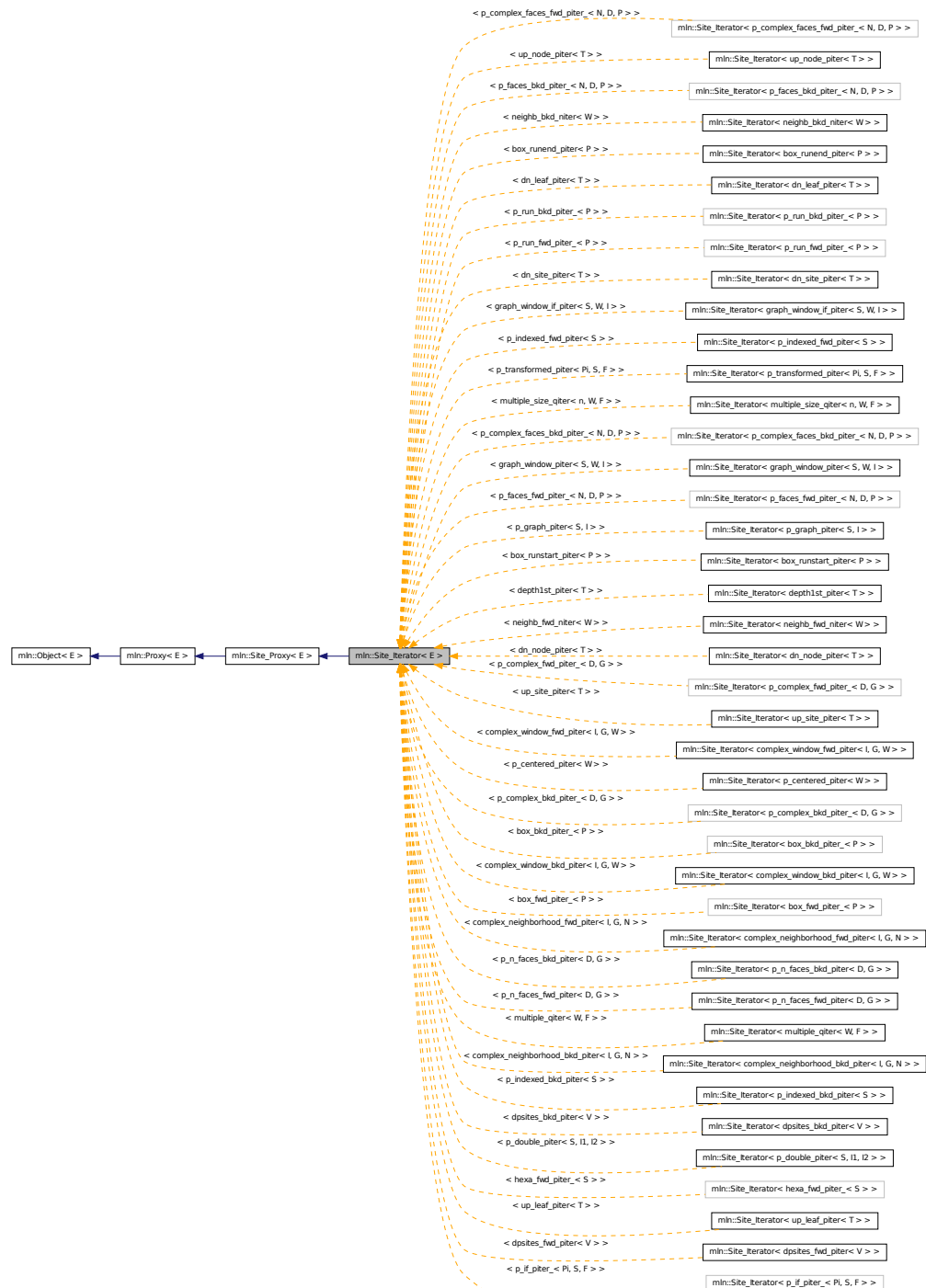
Definition at line 46 of file site.hh.

10.312 mln::Site_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on points.

```
#include <site_iterator.hh>
```

Inheritance diagram for `min::Site_Iterator< E >`:



Public Member Functions

- void [next](#) ()
Go to the next element.

10.312.1 Detailed Description

template<typename E> struct mln::Site_Iterator< E >

Base class for implementation of classes of iterator on points. An iterator on points is an iterator that browse over a set of points.

See also

[mln::doc::Site_Iterator](#) for a complete documentation of this class contents.

Definition at line 53 of file site_iterator.hh.

10.312.2 Member Function Documentation

10.312.2.1 template<typename E > void mln::Site_Iterator< E >::next () [inline]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

Definition at line 92 of file site_iterator.hh.

10.313 mln::Site_Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "site proxy".

`#include <site_proxy.hh>`

Inherits [mln::Proxy< E >](#).

Inherited by [mln::Pseudo_Site< E >](#), and [mln::Site_Iterator< E >](#).

10.313.1 Detailed Description

template<typename E> struct mln::Site_Proxy< E >

Base class for implementation classes of the notion of "site proxy". FIXME: Explain...

Definition at line 61 of file site_proxy.hh.

10.314 mln::Site_Proxy< void > Struct Template Reference

[Site_Proxy](#) category flag type.

```
#include <site_proxy.hh>
```

10.314.1 Detailed Description

template<> struct mln::Site_Proxy< void >

[Site_Proxy](#) category flag type.

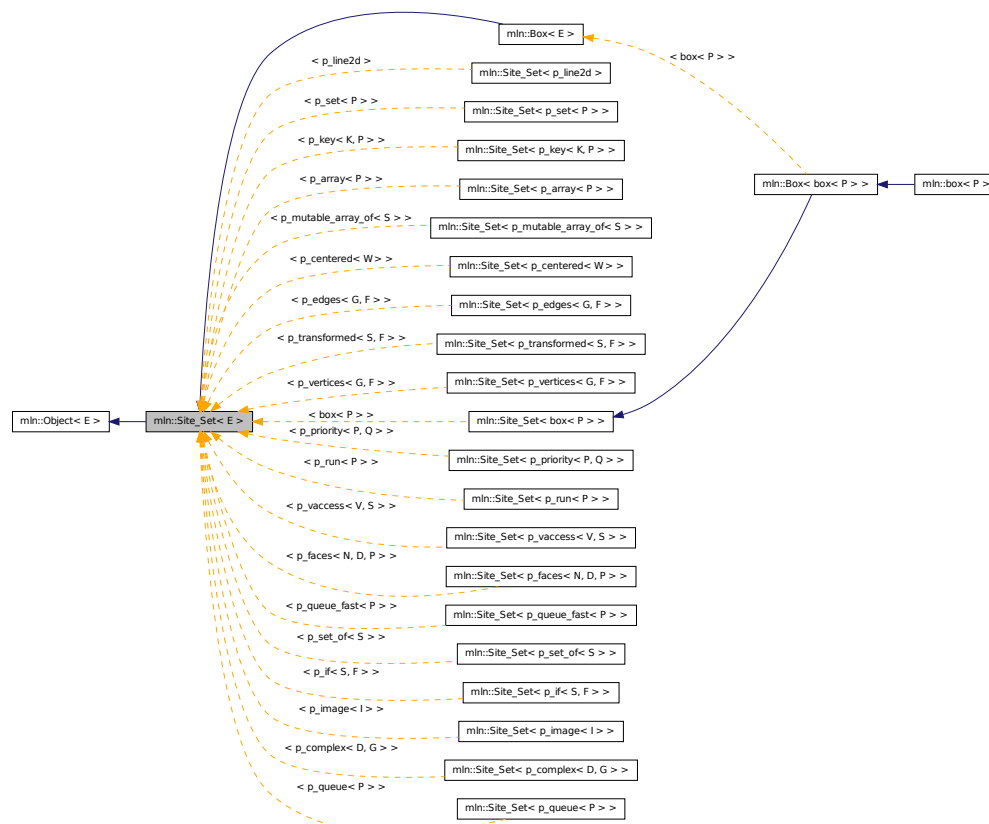
Definition at line 49 of file site_proxy.hh.

10.315 mln::Site_Set< E > Struct Template Reference

Base class for implementation classes of site sets.

```
#include <site_set.hh>
```

Inheritance diagram for mln::Site_Set< E >:



Related Functions

(Note that these are not member functions.)

- `template<typename SI, typename Sr >`
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of point sets.
- `template<typename SI, typename Sr >`
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Strict inclusion test between site sets lhs and rhs.
- `template<typename S >`
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`
Print a site set set into the output stream ostr.
- `template<typename SI, typename Sr >`
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename SI, typename Sr >`
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename SI, typename Sr >`
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Union of a couple of point sets.
- `template<typename S >`
`p_set< typename S::site > unique (const Site_Set< S > &s)`
Give the unique set of s.

10.315.1 Detailed Description

`template<typename E> struct mln::Site_Set< E >`

Base class for implementation classes of site sets.

See also

[mln::doc::Site_Set](#) for a complete documentation of this class contents.

Definition at line 65 of file `mln/core/concept/site_set.hh`.

10.315.2 Friends And Related Function Documentation

10.315.2.1 `template<typename SI, typename Sr > p_set< typename SI::site > diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

10.315.2.2 `template<typename SI, typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

10.315.2.3 `template<typename SI, typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

10.315.2.4 `template<typename S > std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set) [related]`

Print a site set `set` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

10.315.2.5 `template<typename SI, typename Sr > bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

Definition at line 491 of file `operators.hh`.

10.315.2.6 `template<typename Sl, typename Sr > bool operator==(const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related]`

Equality test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set.

[in] *rhs* Another site set.

Definition at line 467 of file operators.hh.

10.315.2.7 `template<typename Sl, typename Sr > p_set< typename Sl::site > sym_diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.315.2.8 `template<typename Sl, typename Sr > p_set< typename Sl::site > uni (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related]`

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.315.2.9 `template<typename S > p_set< typename S::site > unique (const Site_Set< S > & s) [related]`

Give the unique set of `s`.

Definition at line 61 of file unique.hh.

10.316 mln::Site_Set< void > Struct Template Reference

[Site_Set](#) category flag type.

```
#include <site_set.hh>
```

10.316.1 Detailed Description

`template<> struct mln::Site_Set< void >`

[Site_Set](#) category flag type.

Definition at line 54 of file mln/core/concept/site_set.hh.

10.317 mln::sub_image< I, S > Class Template Reference

[Image](#) having its domain restricted by a site set.

```
#include <sub_image.hh>
```

Inherits `image_domain_morpher< I, S, sub_image< I, S > >`.

Public Types

- typedef `sub_image< tag::image_< I >, tag::domain_< S > >` `skeleton`
Skeleton.

Public Member Functions

- `const S & domain () const`
Give the definition domain.
- `operator sub_image< const I, S > () const`
Const promotion via conversion.
- `sub_image ()`
Constructor without argument.
- `sub_image (const I &ima, const S &pset)`
Constructor.

10.317.1 Detailed Description

```
template<typename I, typename S> class mln::sub_image< I, S >
```

`Image` having its domain restricted by a site set.

Definition at line 102 of file `sub_image.hh`.

10.317.2 Member Typedef Documentation

10.317.2.1 `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

Definition at line 108 of file `sub_image.hh`.

10.317.3 Constructor & Destructor Documentation

10.317.3.1 `template<typename I, typename S > mln::sub_image< I, S >::sub_image ()`
`[inline]`

Constructor without argument.

Definition at line 182 of file `sub_image.hh`.

10.317.3.2 `template<typename I, typename S > mln::sub_image< I, S >::sub_image (const I & ima, const S & pset) [inline]`

Constructor.

Definition at line 188 of file sub_image.hh.

10.317.4 Member Function Documentation

10.317.4.1 `template<typename I, typename S > const S & mln::sub_image< I, S >::domain () const [inline]`

Give the definition domain.

Definition at line 205 of file sub_image.hh.

10.317.4.2 `template<typename I, typename S > mln::sub_image< I, S >::operator sub_image< const I, S > () const [inline]`

Const promotion via conversion.

Definition at line 212 of file sub_image.hh.

10.318 mln::sub_image_if< I, S > Struct Template Reference

[Image](#) having its domain restricted by a site set and a function.

```
#include <sub_image_if.hh>
```

Inherits `image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >`.

Public Types

- typedef `sub_image_if< tag::image_< I >, tag::domain_< S > >` [skeleton](#)
Skeleton.

Public Member Functions

- const `p_if< S, fun::p2b::has< I > > & domain ()` const
Give the definition domain.
- `sub_image_if ()`
Constructor without argument.
- `sub_image_if (I &ima, const S &s)`
Constructor.

10.318.1 Detailed Description

`template<typename I, typename S> struct mln::sub_image_if< I, S >`

[Image](#) having its domain restricted by a site set and a function.

Definition at line 101 of file `sub_image_if.hh`.

10.318.2 Member Typedef Documentation

10.318.2.1 `template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton`

Skeleton.

Definition at line 106 of file `sub_image_if.hh`.

10.318.3 Constructor & Destructor Documentation

10.318.3.1 `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if () [inline]`

Constructor without argument.

Definition at line 181 of file `sub_image_if.hh`.

10.318.3.2 `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if (I & ima, const S & s) [inline]`

Constructor.

Definition at line 187 of file `sub_image_if.hh`.

10.318.4 Member Function Documentation

10.318.4.1 `template<typename I, typename S > const p_if< S, fun::p2b::has< I > > & mln::sub_image_if< I, S >::domain () const [inline]`

Give the definition domain.

Definition at line 204 of file `sub_image_if.hh`.

10.319 mln::thru_image< I, F > Class Template Reference

Morph image values through a function.

`#include <thru_image.hh>`

Public Member Functions

- `operator thru_image< const I, F > () const`

Const promotion via conversion.

10.319.1 Detailed Description

template<typename I, typename F> class mln::thru_image< I, F >

Morph image values through a function.

Definition at line 156 of file thru_image.hh.

10.319.2 Member Function Documentation

10.319.2.1 template<typename I, typename F > mln::thru_image< I, F >::operator thru_image< const I, F > () const [inline]

Const promotion via conversion.

Definition at line 239 of file thru_image.hh.

10.320 mln::thrubin_image< I1, I2, F > Class Template Reference

Morphes values from two images through a binary function.

#include <thrubin_image.hh>

Inherits image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >.

Public Types

- typedef I1::psite [psite](#)
Point_Site associated type.
- typedef [value](#) [rvalue](#)
Return type of read-only access.
- typedef [thrubin_image](#)< tag::image_< I1 >, tag::image_< I2 >, F > [skeleton](#)
Skeleton.
- typedef F::result [value](#)
Value associated type.

Public Member Functions

- [operator thrubin_image](#)< const I1, const I2, F > () const
Const promotion via conversion.

10.320.1 Detailed Description

template<typename I1, typename I2, typename F> class mln::thrubin_image< I1, I2, F >

Morphes values from two images through a binary function.

Definition at line 82 of file thrubin_image.hh.

10.320.2 Member Typedef Documentation

**10.320.2.1 template<typename I1, typename I2, typename F> typedef I1 ::psite
 mln::thrubin_image< I1, I2, F >::psite**

Point_Site associated type.

Definition at line 94 of file thrubin_image.hh.

**10.320.2.2 template<typename I1, typename I2, typename F> typedef value
 mln::thrubin_image< I1, I2, F >::rvalue**

Return type of read-only access.

Definition at line 100 of file thrubin_image.hh.

**10.320.2.3 template<typename I1, typename I2, typename F> typedef thrubin_
 image<tag::image_<I1>, tag::image_<I2>, F> mln::thrubin_image< I1, I2, F
 >::skeleton**

Skeleton.

Definition at line 91 of file thrubin_image.hh.

**10.320.2.4 template<typename I1, typename I2, typename F> typedef F ::result
 mln::thrubin_image< I1, I2, F >::value**

[Value](#) associated type.

Definition at line 97 of file thrubin_image.hh.

10.320.3 Member Function Documentation

**10.320.3.1 template<typename I1 , typename I2 , typename F > mln::thrubin_image< I1, I2, F
 >::operator thrubin_image< const I1, const I2, F > () const [inline]**

Const promotion via conversion.

Definition at line 186 of file thrubin_image.hh.

10.321 mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_dim_connected_n_face_bkd_iter](#) ()
Construction.

10.321.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >
```

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj_higher_dim_connected_n_face_iter.hh.

10.321.2 Constructor & Destructor Documentation

10.321.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter () [inline]`

Construction.

Definition at line 196 of file adj_higher_dim_connected_n_face_iter.hh.

10.321.3 Member Function Documentation

10.321.3.1 `void mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.322 mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >, and mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_dim_connected_n_face_fwd_iter](#) ()
Construction.

10.322.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >
```

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj_higher_dim_connected_n_face_iter.hh.

10.322.2 Constructor & Destructor Documentation

10.322.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter () [inline]`

Construction.

Definition at line 162 of file adj_higher_dim_connected_n_face_iter.hh.

10.322.3 Member Function Documentation

10.322.3.1 void mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >::next ()
[inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.323 mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_higher_face_bkd_iter](#) ()

Construction.

10.323.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_face_bkd_iter< D >
```

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 106 of file adj_higher_face_iter.hh.

10.323.2 Constructor & Destructor Documentation

10.323.2.1 `template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter () [inline]`

Construction.

Definition at line 167 of file adj_higher_face_iter.hh.

10.323.3 Member Function Documentation

10.323.3.1 `void mln::Iterator< adj_higher_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.324 mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_higher_face_fwd_iter](#) ()

Construction.

10.324.1 Detailed Description

`template<unsigned D> class mln::topo::adj_higher_face_fwd_iter< D >`

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 72 of file adj_higher_face_iter.hh.

10.324.2 Constructor & Destructor Documentation

10.324.2.1 `template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter() [inline]`

Construction.

Definition at line 139 of file adj_higher_face_iter.hh.

10.324.3 Member Function Documentation

10.324.3.1 `void mln::Iterator< adj_higher_face_fwd_iter< D > >::next() [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.325 mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_lower_dim_connected_n_face_bkd_iter](#) ()

Construction.

10.325.1 Detailed Description

template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj_lower_dim_connected_n_face_iter.hh.

10.325.2 Constructor & Destructor Documentation

10.325.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter () [inline]

Construction.

Definition at line 196 of file adj_lower_dim_connected_n_face_iter.hh.

10.325.3 Member Function Documentation

10.325.3.1 void mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.326 mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >, and mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_dim_connected_n_face_fwd_iter](#) ()
Construction.

10.326.1 Detailed Description

template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj_lower_dim_connected_n_face_iter.hh.

10.326.2 Constructor & Destructor Documentation

10.326.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter () [inline]

Construction.

Definition at line 162 of file adj_lower_dim_connected_n_face_iter.hh.

10.326.3 Member Function Documentation

10.326.3.1 void mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.327 mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits `backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >`.

Public Member Functions

- `void next()`
Go to the next element.
- `adj_lower_face_bkd_iter()`
Construction.

10.327.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_face_bkd_iter< D >
```

Backward iterator on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 108 of file `adj_lower_face_iter.hh`.

10.327.2 Constructor & Destructor Documentation

10.327.2.1 `template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter() [inline]`

Construction.

Definition at line 169 of file `adj_lower_face_iter.hh`.

10.327.3 Member Function Documentation

10.327.3.1 `void mln::Iterator< adj_lower_face_bkd_iter< D > >::next() [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.328 mln::topo::adj_lower_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_face_fwd_iter](#) ()
Construction.

10.328.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 73 of file adj_lower_face_iter.hh.

10.328.2 Constructor & Destructor Documentation

10.328.2.1 `template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter () [inline]`

Construction.

Definition at line 141 of file adj_lower_face_iter.hh.

10.328.3 Member Function Documentation

10.328.3.1 `void mln::Iterator< adj_lower_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.329 mln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_bkd_iter](#) ()
Construction.

10.329.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_higher_face_bkd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 102 of file adj_lower_higher_face_iter.hh.

10.329.2 Constructor & Destructor Documentation

10.329.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter () [inline]`

Construction.

Definition at line 152 of file adj_lower_higher_face_iter.hh.

10.329.3 Member Function Documentation

10.329.3.1 `void mln::Iterator< adj_lower_higher_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.330 mln::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D >, adj_lower_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_lower_higher_face_fwd_iter](#) ()

Construction.

10.330.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_higher_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 71 of file adj_lower_higher_face_iter.hh.

10.330.2 Constructor & Destructor Documentation

10.330.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter () [inline]`

Construction.

Definition at line 133 of file adj_lower_higher_face_iter.hh.

10.330.3 Member Function Documentation

10.330.3.1 `void mln::Iterator< adj_lower_higher_face_fwd_iter< D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.331 `mln::topo::adj_m_face_bkd_iter< D >` Class Template Reference

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits `backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D >>`, and `mln::topo::internal::adj_m_face_iterator< D >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `adj_m_face_bkd_iter ()`
Construction.
- `template<typename Fref > adj_m_face_bkd_iter (const Fref &f_ref, unsigned m)`
Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.331.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_m_face_bkd_iter< D >
```

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

Template Parameters

D The dimension of the complex this iterator belongs to.

The dimension parameter (*m_*) must be lower or equal to D.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 118 of file `adj_m_face_iter.hh`.

10.331.2 Constructor & Destructor Documentation

10.331.2.1 `template<unsigned D> mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter () [inline]`

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 223 of file adj_m_face_iter.hh.

10.331.2.2 `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter (const Fref & f_ref, unsigned m) [inline]`

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 230 of file adj_m_face_iter.hh.

10.331.3 Member Function Documentation

10.331.3.1 `void mln::Iterator< adj_m_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.332 mln::topo::adj_m_face_fwd_iter< D > Class Template Reference

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >, and mln::topo::internal::adj_m_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_m_face_fwd_iter](#) ()
Construction.

- `template<typename Fref >`
`adj_m_face_fwd_iter` (const Fref &f_ref, unsigned m)
Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.332.1 Detailed Description

`template<unsigned D> class mln::topo::adj_m_face_fwd_iter< D >`

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

Template Parameters

D The dimension of the complex this iterator belongs to.

The dimension parameter (*m_*) must be lower or equal to *D*.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 70 of file `adj_m_face_iter.hh`.

10.332.2 Constructor & Destructor Documentation

10.332.2.1 `template<unsigned D> mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter () [inline]`

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 194 of file `adj_m_face_iter.hh`.

10.332.2.2 `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter (const Fref & f_ref, unsigned m) [inline]`

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 201 of file `adj_m_face_iter.hh`.

10.332.3 Member Function Documentation

10.332.3.1 `void mln::Iterator< adj_m_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

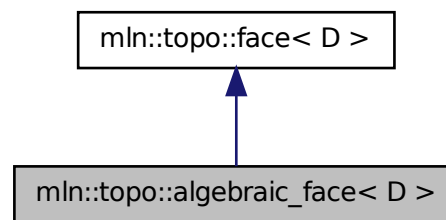
The iterator is valid.

10.333 mln::topo::algebraic_face< D > Class Template Reference

Algebraic face handle in a complex; the face dimension is dynamic.

```
#include <algebraic_face.hh>
```

Inheritance diagram for mln::topo::algebraic_face< D >:



Public Member Functions

- [algebraic_face](#) ()
Build a non-initialized algebraic face handle.
- [algebraic_face](#) ([complex](#)< D > &[complex](#), unsigned n, unsigned face_id, bool [sign](#))
Build an algebraic face handle from complex and face_id.
- [template](#)<unsigned N>
[algebraic_face](#) (const [algebraic_n_face](#)< N, D > &f)
Build a face handle from an [mln::topo::algebraic_n_face](#).
- [algebraic_face](#) (const [face](#)< D > &f, bool [sign](#))
Build an algebraic face handle from an mln::face.
- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?
- bool [sign](#) () const
Accessors.
- void [set_sign](#) (bool [sign](#))
Set the sign of this face.

- `complex< D > cplx () const`
Accessors.
- `unsigned n () const`
Return the dimension of the face.
- `unsigned face_id () const`
Return the id of the face.
- `void set_cplx (const complex< D > &cplx)`
Set the complex the face belongs to.
- `void set_n (unsigned n)`
Set the dimension of the face.
- `void inc_n ()`
Increment the dimension of the face.
- `void dec_n ()`
Decrement the dimension of the face.
- `void set_face_id (unsigned face_id)`
Set the id of the face.
- `void inc_face_id ()`
Increment the id of the face.
- `void dec_face_id ()`
Decrement the id of the face.
- `template<unsigned N>
face_data< N, D > & data () const`
Return the mln::topo::face_data pointed by this handle.
- `std::vector< algebraic_face< D > > lower_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n-1)-faces.
- `std::vector< algebraic_face< D > > higher_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n+1)-faces.

10.333.1 Detailed Description

template<unsigned D> class mln::topo::algebraic_face< D >

Algebraic face handle in a complex; the face dimension is dynamic. Contrary to an `mln::topo::algebraic_n_face`, the dimension of an `mln::topo::algebraic_face` is not fixed.

Definition at line 60 of file `algebraic_face.hh`.

10.333.2 Constructor & Destructor Documentation

10.333.2.1 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face ()`
`[inline]`

Build a non-initialized algebraic face handle.

Definition at line 157 of file algebraic_face.hh.

10.333.2.2 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face (complex<`
`D > & complex, unsigned n, unsigned face_id, bool sign) [inline]`

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 164 of file algebraic_face.hh.

10.333.2.3 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face (const face<`
`D > & f, bool sign) [inline]`

Build an algebraic face handle from an mln::face.

Definition at line 174 of file algebraic_face.hh.

References mln::topo::face< D >::n().

10.333.2.4 `template<unsigned D> template<unsigned N> mln::topo::algebraic_face< D`
`>::algebraic_face (const algebraic_n_face< N, D > & f) [inline]`

Build a face handle from an [mln::topo::algebraic_n_face](#).

Definition at line 184 of file algebraic_face.hh.

10.333.3 Member Function Documentation

10.333.3.1 `template<unsigned D> complex< D > mln::topo::face< D >::cplx () const`
`[inline, inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by mln::complex_psite< D, G >::complex_psite(), mln::topo::operator!=(), and mln::topo::operator==().

10.333.3.2 `template<unsigned D> template<unsigned N> face_data< N, D > &`
`mln::topo::face< D >::data () const [inline, inherited]`

Return the mln::topo::face_data pointed by this handle.

Definition at line 305 of file face.hh.

References mln::topo::face< D >::is_valid().

10.333.3.3 `template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline, inherited]`

Decrement the id of the face.

Definition at line 296 of file face.hh.

10.333.3.4 `template<unsigned D> void mln::topo::face< D >::dec_n () [inline, inherited]`

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

10.333.3.5 `template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline, inherited]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by `mln::geom::complex_geometry< D, P >::operator()()`, and `mln::topo::operator==()`.

10.333.3.6 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces () const [inline, inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.333.3.7 `template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline, inherited]`

Increment the id of the face.

Definition at line 288 of file face.hh.

10.333.3.8 `template<unsigned D> void mln::topo::face< D >::inc_n () [inline, inherited]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.333.3.9 `template<unsigned D> void mln::topo::face< D >::invalidate () [inline, inherited]`

Invalidate this handle.

Definition at line 215 of file face.hh.

References `mln::topo::face< D >::set_face_id()`, and `mln::topo::face< D >::set_n()`.

10.333.3.10 `template<unsigned D> bool mln::topo::face< D >::is_valid () const [inline, inherited]`

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by mln::topo::face< D >::data().

10.333.3.11 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces () const [inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.333.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const [inline, inherited]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by mln::topo::algebraic_face< D >::algebraic_face(), mln::geom::complex_geometry< D, P >::operator()(), and mln::topo::operator==().

10.333.3.13 `template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & cplx) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.333.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id) [inline, inherited]`

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by mln::topo::face< D >::invalidate().

10.333.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n) [inline, inherited]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by mln::topo::face< D >::invalidate().

10.333.3.16 `template<unsigned D> void mln::topo::algebraic_face< D >::set_sign (bool sign) [inline]`

Set the sign of this face.

Definition at line 203 of file algebraic_face.hh.

10.333.3.17 `template<unsigned D> bool mln::topo::algebraic_face< D >::sign () const`
`[inline]`

Accessors.

Return the sign of this face.

Definition at line 195 of file algebraic_face.hh.

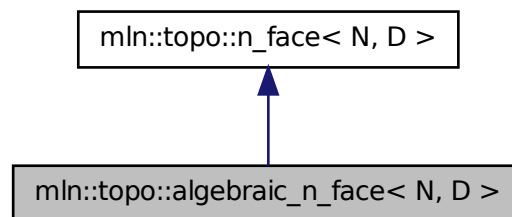
Referenced by `mln::topo::operator==()`.

10.334 mln::topo::algebraic_n_face< N, D > Class Template Reference

Algebraic N-face handle in a complex.

```
#include <algebraic_n_face.hh>
```

Inheritance diagram for `mln::topo::algebraic_n_face< N, D >`:



Public Member Functions

- `algebraic_n_face ()`
Build a non-initialized algebraic face handle.
- `algebraic_n_face (complex< D > &complex, unsigned face_id, bool sign)`
Build an algebraic face handle from complex and face_id.
- `algebraic_n_face (const n_face< N, D > &f, bool sign)`
Build an algebraic face handle from an mln::n_face.
- `void invalidate ()`
Invalidate this handle.

- bool [is_valid](#) () const
Is this handle valid?
- bool [sign](#) () const
Accessors.
- void [set_sign](#) (bool [sign](#))
Set the sign of this face.
- [complex](#)< D > [cplx](#) () const
Accessors.
- unsigned [face_id](#) () const
Return the id of the face.
- void [set_cplx](#) (const [complex](#)< D > &cplx)
Set the complex the face belongs to.
- unsigned [n](#) () const
Return the dimension of the face.
- void [set_face_id](#) (unsigned face_id)
Set the id of the face.
- void [inc_face_id](#) ()
Increment the id of the face.
- void [dec_face_id](#) ()
Decrement the id of the face.
- [face_data](#)< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_n_face](#)< N-1, D > > [lower_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_n_face](#)< N+1, D > > [higher_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.334.1 Detailed Description

template<unsigned N, unsigned D> class mln::topo::algebraic_n_face< N, D >

Algebraic N-face handle in a complex. Contrary to an [mln::topo::algebraic_face](#), the dimension of an [mln::topo::algebraic_n_face](#) is fixed.

Definition at line 50 of file algebraic_n_face.hh.

10.334.2 Constructor & Destructor Documentation

10.334.2.1 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face () [inline]`

Build a non-initialized algebraic face handle.

Definition at line 166 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

10.334.2.2 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (complex< D > & complex, unsigned face_id, bool sign) [inline]`

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 176 of file `algebraic_n_face.hh`.

10.334.2.3 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (const n_face< N, D > & f, bool sign) [inline]`

Build an algebraic face handle from an `mln::n_face`.

Definition at line 186 of file `algebraic_n_face.hh`.

10.334.3 Member Function Documentation

10.334.3.1 `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx () const [inline, inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file `n_face.hh`.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.334.3.2 `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data () const [inline, inherited]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

10.334.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id () [inline, inherited]`

Decrement the id of the face.

Definition at line 243 of file `n_face.hh`.

10.334.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id ()
const [inline, inherited]`

Return the id of the face.

Definition at line 211 of file n_face.hh.

Referenced by mln::topo::operator==().

10.334.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D >
> mln::topo::n_face< N, D >::higher_dim_adj_faces () const [inline,
inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n_face.hh.

References mln::topo::n_face< N, D >::is_valid().

Referenced by mln::topo::edge().

10.334.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ()
[inline, inherited]`

Increment the id of the face.

Definition at line 235 of file n_face.hh.

10.334.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ()
[inline, inherited]`

Invalidate this handle.

Definition at line 187 of file n_face.hh.

References mln::topo::n_face< N, D >::set_face_id().

10.334.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ()
const [inline, inherited]`

Is this handle valid?

Definition at line 179 of file n_face.hh.

Referenced by mln::topo::algebraic_n_face< N, D >::algebraic_n_face(), mln::topo::n_face< N, D >::data(), mln::topo::n_face< N, D >::higher_dim_adj_faces(), mln::topo::n_face< N, D >::lower_dim_adj_faces(), and mln::topo::n_face< N, D >::n_face().

10.334.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D
> > mln::topo::n_face< N, D >::lower_dim_adj_faces () const [inline,
inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file n_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

10.334.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ()
const [inline, inherited]`

Return the dimension of the face.

Definition at line 203 of file `n_face.hh`.

10.334.3.11 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx (const
complex< D > & cplx) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 219 of file `n_face.hh`.

10.334.3.12 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (unsigned
face_id) [inline, inherited]`

Set the id of the face.

Definition at line 227 of file `n_face.hh`.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

10.334.3.13 `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign (bool
sign) [inline]`

Set the sign of this face.

Definition at line 205 of file `algebraic_n_face.hh`.

10.334.3.14 `template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign
() const [inline]`

Accessors.

Return the sign of this face.

Definition at line 197 of file `algebraic_n_face.hh`.

Referenced by `mln::topo::operator==()`.

10.335 mln::topo::center_only_iter< D > Class Template Reference

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

`#include <center_only_iter.hh>`

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_iter< D > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [center_only_iter](#) ()
Construction.

10.335.1 Detailed Description

`template<unsigned D> class mln::topo::center_only_iter< D >`

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

[mln::topo::center_only_iter](#) inherits from `mln::topo::internal::forward_complex_relative_iterator_base`, but it could inherit from `mln::topo::internal::backward_complex_relative_iterator_base` as well, since it always contains a single element, the center/reference face (and the traversal order is meaningless).

This iterator is essentially used to implement other iterators.

See also

`mln::topo::centered_iter_adapter`
`mln::complex_lower_window`
`mln::complex_higher_window`
`mln::complex_lower_higher_window`

Definition at line 73 of file `center_only_iter.hh`.

10.335.2 Constructor & Destructor Documentation

10.335.2.1 `template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter ()`
[inline]

Construction.

Definition at line 107 of file `center_only_iter.hh`.

10.335.3 Member Function Documentation

10.335.3.1 `void mln::Iterator< center_only_iter< D > >::next ()` **[inherited]**

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.336 mln::topo::centered_bkd_iter_adapter< D, I > Class Template Reference

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_bkd_iter_adapter](#) ()
Construction.

10.336.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_bkd_iter_adapter< D, I >
```

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

Definition at line 91 of file centered_iter_adapter.hh.

10.336.2 Constructor & Destructor Documentation

10.336.2.1 `template<unsigned D, typename I > mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter () [inline]`

Construction.

Definition at line 141 of file centered_iter_adapter.hh.

10.336.3 Member Function Documentation

10.336.3.1 `void mln::Iterator< centered_bkd_iter_adapter< D, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.337 mln::topo::centered_fwd_iter_adapter< D, I > Class Template Reference

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_fwd_iter_adapter](#) ()
Construction.

10.337.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_fwd_iter_adapter< D, I >
```

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

Definition at line 57 of file centered_iter_adapter.hh.

10.337.2 Constructor & Destructor Documentation

10.337.2.1 `template<unsigned D, typename I > mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter () [inline]`

Construction.

Definition at line 122 of file centered_iter_adapter.hh.

10.337.3 Member Function Documentation

10.337.3.1 void mln::Iterator< centered_fwd_iter_adapter< D, I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.338 mln::topo::complex< D > Class Template Reference

General complex of dimension D.

```
#include <complex.hh>
```

Public Types

- typedef [face_bkd_iter](#)< D > [bkd_citer](#)
Backward [mln::Iterator](#) type iterating on all faces.
- typedef [face_fwd_iter](#)< D > [fwd_citer](#)
Forward [mln::Iterator](#) type iterating on all faces.

Public Member Functions

- const void * [addr](#) () const
Get the address of the data of this complex.
- [complex](#) ()
Complex construction.
- [n_face](#)< 0u, D > [add_face](#) ()
Add a 0-face to the complex.
- template<unsigned N>
[n_face](#)< N+1, D > [add_face](#) (const [n_faces_set](#)< N, D > &adjacent_faces)
Add a (N+1)-face to the complex (with N >= 0).
- unsigned [nfaces](#) () const
Static manipulators.

- template<unsigned N>
unsigned [nfaces_of_static_dim](#) () const
Return the number of N -faces.
- unsigned [nfaces_of_dim](#) (unsigned n) const
Dynamic manipulators.
- void [print](#) (std::ostream &ostr) const
Pretty-printing.
- template<unsigned N>
void [print_faces](#) (std::ostream &ostr) const
Print the faces of dimension N .

10.338.1 Detailed Description

template<unsigned D> class mln::topo::complex< D >

General complex of dimension D.

Definition at line 87 of file complex.hh.

10.338.2 Member Typedef Documentation

10.338.2.1 template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer

Backward [mln::Iterator](#) type iterating on all faces.

Definition at line 93 of file complex.hh.

10.338.2.2 template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer

Forward [mln::Iterator](#) type iterating on all faces.

Definition at line 91 of file complex.hh.

10.338.3 Constructor & Destructor Documentation

10.338.3.1 template<unsigned D> mln::topo::complex< D >::complex () [inline]

Complex construction.

Create a new `D-complex`.

Definition at line 471 of file complex.hh.

10.338.4 Member Function Documentation

10.338.4.1 `template<unsigned D> n_face< 0u, D > mln::topo::complex< D >::add_face ()`
`[inline]`

Add a 0-face to the complex.

Definition at line 480 of file complex.hh.

10.338.4.2 `template<unsigned D> template<unsigned N> n_face< N+1, D >`
`mln::topo::complex< D >::add_face (const n_faces_set< N, D > & adjacent_faces)`
`[inline]`

Add a (N+1)-face to the complex (with $N \geq 0$).

Parameters

adjacent_faces The (N-1)-faces adjacent to the new N-face.

Definition at line 493 of file complex.hh.

References `mln::topo::n_faces_set< N, D >::faces()`.

10.338.4.3 `template<unsigned D> const void * mln::topo::complex< D >::addr () const`
`[inline]`

Get the address of the data of this complex.

This address is a concise and useful information to print and track the actual content of this complex.

Definition at line 699 of file complex.hh.

10.338.4.4 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces () const`
`[inline]`

Static manipulators.

These methods use statically-known input.

Return the total number of faces, whatever their dimension.

Definition at line 579 of file complex.hh.

10.338.4.5 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces_of_dim (`
`unsigned n) const [inline]`

Dynamic manipulators.

These methods use input know as run time.

Return the number of *n-faces*.

Warning, this function has a complexity linear in term of N, since each `n_faces_set` is checked (the present implementation does not provide a direct access to `n_faces_set` through a dynamic value of the dimension).

Definition at line 601 of file complex.hh.

10.338.4.6 `template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D >::nfaces_of_static_dim () const [inline]`

Return the number of N-faces.

Definition at line 588 of file complex.hh.

10.338.4.7 `template<unsigned D> void mln::topo::complex< D >::print (std::ostream & ostr) const [inline]`

Pretty-printing.

Print the complex.

Definition at line 679 of file complex.hh.

Referenced by mln::topo::operator<<().

10.338.4.8 `template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces (std::ostream & ostr) const [inline]`

Print the faces of dimension N.

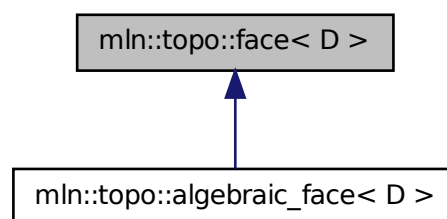
Definition at line 688 of file complex.hh.

10.339 mln::topo::face< D > Class Template Reference

Face handle in a complex; the face dimension is dynamic.

`#include <face.hh>`

Inheritance diagram for mln::topo::face< D >:



Public Member Functions

- [face \(\)](#)

Build a non-initialized face handle.

- `face (complex< D > &complex, unsigned n, unsigned face_id)`
Build a face handle from complex and face_id.
- `template<unsigned N>`
`face (const n_face< N, D > &f)`
Build a face handle from an `mln::topo::n_face`.
- `void invalidate ()`
Invalidate this handle.
- `bool is_valid () const`
Is this handle valid?
- `complex< D > cplx () const`
Accessors.
- `unsigned n () const`
Return the dimension of the face.
- `unsigned face_id () const`
Return the id of the face.
- `void set_cplx (const complex< D > &cplx)`
Set the complex the face belongs to.
- `void set_n (unsigned n)`
Set the dimension of the face.
- `void inc_n ()`
Increment the dimension of the face.
- `void dec_n ()`
Decrement the dimension of the face.
- `void set_face_id (unsigned face_id)`
Set the id of the face.
- `void inc_face_id ()`
Increment the id of the face.
- `void dec_face_id ()`
Decrement the id of the face.
- `template<unsigned N>`
`face_data< N, D > & data () const`
Return the `mln::topo::face_data` pointed by this handle.
- `std::vector< algebraic_face< D > > lower_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n-1)-faces.
- `std::vector< algebraic_face< D > > higher_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n+1)-faces.

10.339.1 Detailed Description

template<unsigned D> class mln::topo::face< D >

Face handle in a complex; the face dimension is dynamic. Contrary to an [mln::topo::n_face](#), the dimension of an [mln::topo::face](#) is not fixed.

Definition at line 64 of file face.hh.

10.339.2 Constructor & Destructor Documentation

10.339.2.1 template<unsigned D> mln::topo::face< D >::face () [inline]

Build a non-initialized face handle.

Definition at line 178 of file face.hh.

10.339.2.2 template<unsigned D> mln::topo::face< D >::face (complex< D > & complex, unsigned n, unsigned face_id) [inline]

Build a face handle from *complex* and *face_id*.

Definition at line 187 of file face.hh.

10.339.2.3 template<unsigned D> template<unsigned N> mln::topo::face< D >::face (const n_face< N, D > & f) [inline]

Build a face handle from an [mln::topo::n_face](#).

Definition at line 197 of file face.hh.

10.339.3 Member Function Documentation

10.339.3.1 template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline]

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.339.3.2 template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data () const [inline]

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file face.hh.

References `mln::topo::face< D >::is_valid()`.

10.339.3.3 **template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline]**

Decrement the id of the face.

Definition at line 296 of file face.hh.

10.339.3.4 **template<unsigned D> void mln::topo::face< D >::dec_n () [inline]**

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

10.339.3.5 **template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline]**

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by mln::geom::complex_geometry< D, P >::operator>(), and mln::topo::operator==().

10.339.3.6 **template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces () const [inline]**

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.339.3.7 **template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline]**

Increment the id of the face.

Definition at line 288 of file face.hh.

10.339.3.8 **template<unsigned D> void mln::topo::face< D >::inc_n () [inline]**

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.339.3.9 **template<unsigned D> void mln::topo::face< D >::invalidate () [inline]**

Invalidate this handle.

Definition at line 215 of file face.hh.

References mln::topo::face< D >::set_face_id(), and mln::topo::face< D >::set_n().

10.339.3.10 **template<unsigned D> bool mln::topo::face< D >::is_valid () const [inline]**

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by mln::topo::face< D >::data().

10.339.3.11 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.339.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const [inline]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by `mln::topo::algebraic_face< D >::algebraic_face()`, `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==()`.

10.339.3.13 `template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & cplx) [inline]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.339.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id) [inline]`

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

10.339.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n) [inline]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

10.340 mln::topo::face_bkd_iter< D > Class Template Reference

Backward iterator on all the faces of an `mln::complex<D>`.

```
#include <face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >`.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [face_bkd_iter\(\)](#)
Construction and assignment.
- `void start()`
Manipulation.

10.340.1 Detailed Description

template<unsigned D> class mln::topo::face_bkd_iter< D >

Backward iterator on all the faces of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 112 of file `face_iter.hh`.

10.340.2 Constructor & Destructor Documentation

10.340.2.1 template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter ()
[inline]

Construction and assignment.

Definition at line 207 of file `face_iter.hh`.

10.340.3 Member Function Documentation

10.340.3.1 void mln::Iterator< face_bkd_iter< D > >::next () **[inherited]**

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.340.3.2 template<unsigned D> void mln::topo::face_bkd_iter< D >::start () **[inline]**

Manipulation.

Start an iteration.

Definition at line 224 of file `face_iter.hh`.

10.341 mln::topo::face_fwd_iter< D > Class Template Reference

Forward iterator on all the faces of an mln::complex<D>.

```
#include <face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [face_fwd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.341.1 Detailed Description

```
template<unsigned D> class mln::topo::face_fwd_iter< D >
```

Forward iterator on all the faces of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 69 of file face_iter.hh.

10.341.2 Constructor & Destructor Documentation

10.341.2.1 `template<unsigned D> mln::topo::face_fwd_iter< D >::face_fwd_iter ()`
`[inline]`

Construction and assignment.

Definition at line 155 of file face_iter.hh.

10.341.3 Member Function Documentation

10.341.3.1 `void mln::Iterator< face_fwd_iter< D > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.341.3.2 `template<unsigned D> void mln::topo::face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

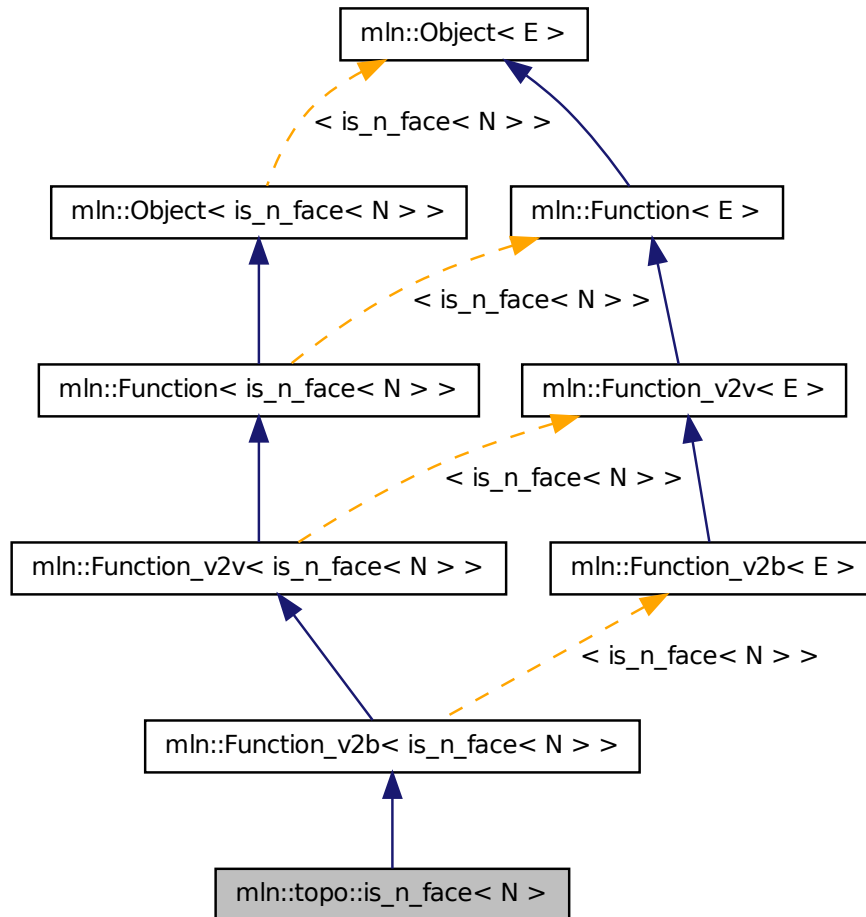
Definition at line 172 of file face_iter.hh.

10.342 `mln::topo::is_n_face< N >` Struct Template Reference

A functor testing wheter a [mln::complex_psite](#) is an N -face.

```
#include <is_n_face.hh>
```

Inheritance diagram for mln::topo::is_n_face< N >:



10.342.1 Detailed Description

```
template<unsigned N> struct mln::topo::is_n_face< N >
```

A functor testing whether a [mln::complex_site](#) is an N-face.

Definition at line 48 of file `is_n_face.hh`.

10.343 mln::topo::is_simple_2d_t< N > Struct Template Reference

Test if a point is simple or not.

```
#include <is_simple_2d.hh>
```

10.343.1 Detailed Description

```
template<typename N> struct mln::topo::is_simple_2d_t< N >
```

Test if a point is simple or not. A point of an object is simple if in its c8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

- - | | P | Here p is simple in the c4 and c8 case. | | |

- | - | P | Here p is never simple. | | |

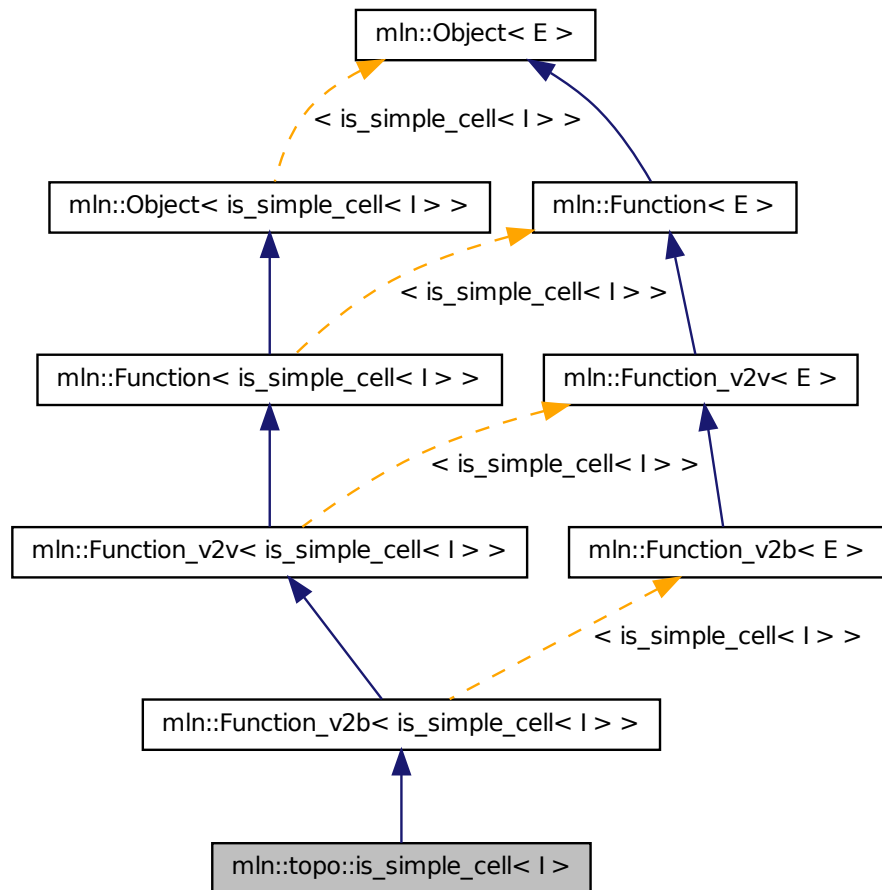
Definition at line 66 of file is_simple_2d.hh.

10.344 mln::topo::is_simple_cell< I > Class Template Reference

A predicate for the simplicity of a point based on the collapse property of the attachment.

```
#include <is_simple_cell.hh>
```

Inheritance diagram for mln::topo::is_simple_cell< I >:



Public Types

- typedef `mln::complex_psite< D, G >` `psite`
Psite type.
- typedef bool `result`
Result type of the functor.

Public Member Functions

- typedef `mln_geom (I) G`
Geometry of the image.

- bool [operator\(\)](#) (const [mln::complex_psite](#)< I::dim, mln_geom(I)> &p) const

Based on the algorithm A2 from couprie.08.pami.

- void [set_image](#) (const [mln::Image](#)< I > &ima)

Set the underlying image.

Static Public Attributes

- static const unsigned [D](#) = I::dim

Dimension of the image (and therefore of the complex).

10.344.1 Detailed Description

template<typename I> class mln::topo::is_simple_cell< I >

A predicate for the simplicity of a point based on the collapse property of the attachment. The functor does not actually take a cell as input, but a face that is expected to be a D-facet.

Definition at line 57 of file is_simple_cell.hh.

10.344.2 Member Typedef Documentation

10.344.2.1 **template<typename I > typedef mln::complex_psite<D, G>
mln::topo::is_simple_cell< I >::psite**

Psite type.

Definition at line 65 of file is_simple_cell.hh.

10.344.2.2 **template<typename I > typedef bool mln::topo::is_simple_cell< I >::result**

Result type of the functor.

Reimplemented from [mln::Function_v2b< is_simple_cell< I > >](#).

Definition at line 68 of file is_simple_cell.hh.

10.344.3 Member Function Documentation

10.344.3.1 **template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom (I)**

Geometry of the image.

10.344.3.2 **template<typename I > bool mln::topo::is_simple_cell< I >::operator() (const
mln::complex_psite< I::dim, mln_geom(I)> & p) const [inline]**

Based on the algorithm A2 from couprie.08.pami.

Definition at line 115 of file is_simple_cell.hh.

References mln::make::attachment().

10.344.3.3 `template<typename I> void mln::topo::is_simple_cell< I >::set_image (const mln::Image< I> & ima) [inline]`

Set the underlying image.

Definition at line 107 of file is_simple_cell.hh.

10.344.4 Member Data Documentation

10.344.4.1 `template<typename I> const unsigned mln::topo::is_simple_cell< I >::D = I::dim [static]`

Dimension of the image (and therefore of the complex).

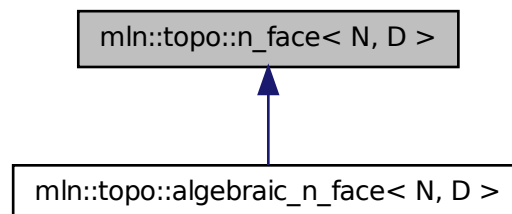
Definition at line 61 of file is_simple_cell.hh.

10.345 mln::topo::n_face< N, D > Class Template Reference

N-face handle in a complex.

`#include <n_face.hh>`

Inheritance diagram for mln::topo::n_face< N, D >:



Public Member Functions

- void `invalidate` ()
Invalidate this handle.
- bool `is_valid` () const
Is this handle valid?
- `n_face` ()
Build a non-initialized face handle.

- `n_face` (`complex< D > &complex`, unsigned `face_id`)
Build a face handle from `complex` and `face_id`.
- `complex< D > cplx ()` const
Accessors.
- unsigned `face_id ()` const
Return the id of the face.
- void `set_cplx` (const `complex< D > &cplx`)
Set the complex the face belongs to.
- unsigned `n ()` const
Return the dimension of the face.
- void `set_face_id` (unsigned `face_id`)
Set the id of the face.
- void `inc_face_id ()`
Increment the id of the face.
- void `dec_face_id ()`
Decrement the id of the face.
- `face_data< N, D > &data ()` const
Return the `mln::topo::face_data` pointed by this handle.
- `std::vector< algebraic_n_face< N-1, D > > lower_dim_adj_faces ()` const
Return an array of face handles pointing to adjacent (n-1)-faces.
- `std::vector< algebraic_n_face< N+1, D > > higher_dim_adj_faces ()` const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.345.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::n_face< N, D >`

`N-face` handle in a complex. Contrary to an `mln::topo::face`, the dimension of an `mln::topo::n_face` is fixed.

Definition at line 61 of file `n_face.hh`.

10.345.2 Constructor & Destructor Documentation

10.345.2.1 `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ()`
`[inline]`

Build a non-initialized face handle.

Definition at line 159 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.2.2 `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face (complex< D > & complex, unsigned face_id) [inline]`

Build a face handle from *complex* and *face_id*.

Definition at line 169 of file n_face.hh.

10.345.3 Member Function Documentation

10.345.3.1 `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx () const [inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file n_face.hh.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.345.3.2 `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data () const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file n_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id () [inline]`

Decrement the id of the face.

Definition at line 243 of file n_face.hh.

10.345.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const [inline]`

Return the id of the face.

Definition at line 211 of file n_face.hh.

Referenced by `mln::topo::operator==()`.

10.345.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > mln::topo::n_face< N, D >::higher_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

10.345.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ()`
`[inline]`

Increment the id of the face.

Definition at line 235 of file `n_face.hh`.

10.345.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ()`
`[inline]`

Invalidate this handle.

Definition at line 187 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::set_face_id()`.

10.345.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ()`
`const [inline]`

Is this handle valid?

Definition at line 179 of file `n_face.hh`.

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, `mln::topo::n_face< N, D >::data()`, `mln::topo::n_face< N, D >::higher_dim_adj_faces()`, `mln::topo::n_face< N, D >::lower_dim_adj_faces()`, and `mln::topo::n_face< N, D >::n_face()`.

10.345.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > >`
`mln::topo::n_face< N, D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ()`
`const [inline]`

Return the dimension of the face.

Definition at line 203 of file `n_face.hh`.

10.345.3.11 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx (const`
`complex< D > & cplx) [inline]`

Set the complex the face belongs to.

Definition at line 219 of file `n_face.hh`.

10.345.3.12 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (`
`unsigned face_id) [inline]`

Set the id of the face.

Definition at line 227 of file n_face.hh.

Referenced by mln::topo::n_face< N, D >::invalidate().

10.346 mln::topo::n_face_bkd_iter< D > Class Template Reference

Backward iterator on all the faces of an mln::complex<D>.

```
#include <n_face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- unsigned [n](#) () const
Accessors.

10.346.1 Detailed Description

```
template<unsigned D> class mln::topo::n_face_bkd_iter< D >
```

Backward iterator on all the faces of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 127 of file n_face_iter.hh.

10.346.2 Member Function Documentation

10.346.2.1 `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n () const`
`[inline]`

Accessors.

Shortcuts to face_'s accessors.

Definition at line 308 of file n_face_iter.hh.

Referenced by mln::topo::n_face_bkd_iter< D >::start().

10.346.2.2 void mln::Iterator< n_face_bkd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.346.2.3 template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start () [inline]

Manipulation.

Start an iteration.

Definition at line 275 of file n_face_iter.hh.

References mln::topo::n_face_bkd_iter< D >::n().

10.347 mln::topo::n_face_fwd_iter< D > Class Template Reference

Forward iterator on all the faces of an mln::complex<D>.

```
#include <n_face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- unsigned [n](#) () const
Accessors.

10.347.1 Detailed Description

```
template<unsigned D> class mln::topo::n_face_fwd_iter< D >
```

Forward iterator on all the faces of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 72 of file n_face_iter.hh.

10.347.2 Member Function Documentation

10.347.2.1 `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n () const [inline]`

Accessors.

Shortcuts to face_'s accessors.

Definition at line 235 of file n_face_iter.hh.

10.347.2.2 `void mln::Iterator< n_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.347.2.3 `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 202 of file n_face_iter.hh.

10.348 mln::topo::n_faces_set< N, D > Class Template Reference

Set of face handles of dimension N.

```
#include <n_faces_set.hh>
```

Public Types

- typedef std::vector< [algebraic_n_face](#)< N, D > > [faces_type](#)

The type of the set of face handles.

Public Member Functions

- void [add](#) (const [algebraic_n_face](#)< N, D > &f)
Append an algebraic face f to the set.
- void [reserve](#) (size_t n)
Reserve n cells in the set.
- const [faces_type](#) & [faces](#) () const
Accessors.

10.348.1 Detailed Description

template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >

Set of face handles of dimension N.

Definition at line 56 of file n_faces_set.hh.

10.348.2 Member Typedef Documentation

10.348.2.1 template<unsigned N, unsigned D> typedef std::vector< algebraic_n_face<N, D> > mln::topo::n_faces_set< N, D >::faces_type

The type of the set of face handles.

Definition at line 70 of file n_faces_set.hh.

10.348.3 Member Function Documentation

10.348.3.1 template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::add (const algebraic_n_face< N, D > & f) [inline]

Append an algebraic face f to the set.

Definition at line 171 of file n_faces_set.hh.

References mln::topo::n_face< N, D >::cplx().

Referenced by mln::topo::operator+(), and mln::topo::operator-().

10.348.3.2 template<unsigned N, unsigned D> const std::vector< algebraic_n_face< N, D > > & mln::topo::n_faces_set< N, D >::faces () const [inline]

Accessors.

Return the set of handles.

Definition at line 190 of file n_faces_set.hh.

Referenced by mln::topo::complex< D >::add_face().

10.348.3.3 `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::reserve (size_t n) [inline]`

Reserve n cells in the set.

This methods does not change the content of *faces_*; it only pre-allocate memory. Method reserve is provided for efficiency purpose, and its use is completely optional.

Definition at line 182 of file `n_faces_set.hh`.

10.349 `mln::topo::skeleton::is_simple_point< N >` Struct Template Reference

```
#include <is_simple_point.hh>
```

10.349.1 Detailed Description

`template<typename N> struct mln::topo::skeleton::is_simple_point< N >`

Tell if a point is simple or not. A point of an object is simple if in its c8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

```
- - |
| P | Here p is simple in the c4 and c8 case.
| | |

- | -
| P | Here p is never simple.
| | |
```

Definition at line 68 of file `is_simple_point.hh`.

10.350 `mln::topo::static_n_face_bkd_iter< N, D >` Class Template Reference

Backward iterator on all the N-faces of a `mln::complex<D>`.

```
#include <static_n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `static_n_face_bkd_iter ()`

Construction and assignment.

- void [start](#) ()

Manipulation.

10.350.1 Detailed Description

template<unsigned N, unsigned D> class mln::topo::static_n_face_bkd_iter< N, D >

Backward iterator on all the `N-faces` of a `mln::complex<D>`.

Template Parameters

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 101 of file `static_n_face_iter.hh`.

10.350.2 Constructor & Destructor Documentation

10.350.2.1 template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter () [inline]

Construction and assignment.

Definition at line 194 of file `static_n_face_iter.hh`.

10.350.3 Member Function Documentation

10.350.3.1 void mln::Iterator< static_n_face_bkd_iter< N, D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.350.3.2 template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start () [inline]

Manipulation.

Start an iteration.

Definition at line 217 of file `static_n_face_iter.hh`.

10.351 mln::topo::static_n_face_fwd_iter< N, D > Class Template Reference

Forward iterator on all the N-faces of a mln::complex<D>.

```
#include <static_n_face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [static_n_face_fwd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.351.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::static_n_face_fwd_iter< N, D >
```

Forward iterator on all the N-faces of a mln::complex<D>.

Template Parameters

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 55 of file static_n_face_iter.hh.

10.351.2 Constructor & Destructor Documentation

10.351.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter () [inline]`

Construction and assignment.

Definition at line 145 of file static_n_face_iter.hh.

10.351.3 Member Function Documentation

10.351.3.1 `void mln::Iterator< static_n_face_fwd_iter< N, D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.351.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 168 of file static_n_face_iter.hh.

10.352 mln::tr_image< S, I, T > Struct Template Reference

Transform an image by a given transformation.

`#include <tr_image.hh>`

Inherits `image_identity< I, S, tr_image< S, I, T > >`.

Public Types

- `typedef I::value lvalue`
Return type of read-write access.
- `typedef I::psite psite`
Point_Site associated type.
- `typedef I::value rvalue`
Return type of read-only access.
- `typedef I::site site`
Site associated type.
- `typedef tr_image< S, tag::image_< I >, T > skeleton`
Skeleton.
- `typedef I::value value`
Value associated type.

Public Member Functions

- `const S & domain () const`
Return the domain morpher.

- bool `has` (const vec_t &v) const
Test if a pixel value is accessible at v.
- bool `is_valid` () const
Test if this image has been initialized.
- I::value `operator()` (const psite &p) const
Read-only access of pixel value at point site p.
- void `set_tr` (T &tr)
Set the transformation.
- const T & `tr` () const
Return the underlying transformation.
- `tr_image` (const S &s, const I &ima, const T &tr)
Constructors.

10.352.1 Detailed Description

template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >

Transform an image by a given transformation.

Definition at line 83 of file tr_image.hh.

10.352.2 Member Typedef Documentation

10.352.2.1 template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue

Return type of read-write access.

Definition at line 101 of file tr_image.hh.

10.352.2.2 template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite

Point_Site associated type.

Definition at line 92 of file tr_image.hh.

10.352.2.3 template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue

Return type of read-only access.

Definition at line 104 of file tr_image.hh.

10.352.2.4 `template<typename S, typename I, typename T> typedef I::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

Definition at line 95 of file `tr_image.hh`.

10.352.2.5 `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

Definition at line 107 of file `tr_image.hh`.

10.352.2.6 `template<typename S, typename I, typename T> typedef I::value mln::tr_image< S, I, T >::value`

[Value](#) associated type.

Definition at line 98 of file `tr_image.hh`.

10.352.3 Constructor & Destructor Documentation

10.352.3.1 `template<typename S, typename I, typename T> mln::tr_image< S, I, T >::tr_image(const S & s, const I & ima, const T & tr) [inline]`

Constructors.

Definition at line 174 of file `tr_image.hh`.

10.352.4 Member Function Documentation

10.352.4.1 `template<typename S, typename I, typename T> const S & mln::tr_image< S, I, T >::domain() const [inline]`

Return the domain morpher.

Definition at line 247 of file `tr_image.hh`.

10.352.4.2 `template<typename S, typename I, typename T> bool mln::tr_image< S, I, T >::has(const vec_t & v) const [inline]`

Test if a pixel value is accessible at `v`.

Definition at line 200 of file `tr_image.hh`.

10.352.4.3 `template<typename S, typename I, typename T> bool mln::tr_image< S, I, T >::is_valid() const [inline]`

Test if this image has been initialized.

Definition at line 191 of file `tr_image.hh`.

10.352.4.4 `template<typename S , typename I , typename T > I::value mln::tr_image< S, I, T >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Mutable access is only OK for reading (not writing).

Definition at line 213 of file tr_image.hh.

10.352.4.5 `template<typename S , typename I , typename T > void mln::tr_image< S, I, T >::set_tr (T & tr) [inline]`

Set the transformation.

Definition at line 231 of file tr_image.hh.

10.352.4.6 `template<typename S , typename I , typename T > const T & mln::tr_image< S, I, T >::tr () const [inline]`

Return the underlying transformation.

Definition at line 239 of file tr_image.hh.

10.353 mln::transformed_image< I, F > Struct Template Reference

[Image](#) having its domain restricted by a site set.

```
#include <transformed_image.hh>
```

Inherits image_domain_morpher< I, p_transformed< I::domain_t, F >, transformed_image< I, F > >.

Public Types

- typedef [transformed_image](#)< tag::image_< I >, tag::function_< F > > [skeleton](#)
Skeleton.

Public Member Functions

- const [p_transformed](#)< typename I::domain_t, F > & [domain](#) () const
Give the definition domain.
- [operator transformed_image](#)< const I, F > () const
Const promotion via conversion.
- internal::morpher_lvalue_< I >::ret [operator](#)() (const typename I::psite &p)
*Read and "write if possible" access of pixel value at point site *p*.*
- I::rvalue [operator](#)() (const typename I::psite &p) const
*Read-only access of pixel value at point site *p*.*

- [transformed_image](#) ()

Constructor without argument.

- [transformed_image](#) (I &ima, const F &f)

Constructor.

10.353.1 Detailed Description

`template<typename I, typename F> struct mln::transformed_image< I, F >`

[Image](#) having its domain restricted by a site set.

Definition at line 95 of file transformed_image.hh.

10.353.2 Member Typedef Documentation

10.353.2.1 `template<typename I, typename F> typedef transformed_image< tag::image_<I>, tag::function_<F> > mln::transformed_image< I, F >::skeleton`

Skeleton.

Definition at line 100 of file transformed_image.hh.

10.353.3 Constructor & Destructor Documentation

10.353.3.1 `template<typename I , typename F > mln::transformed_image< I, F >::transformed_image () [inline]`

Constructor without argument.

Definition at line 186 of file transformed_image.hh.

10.353.3.2 `template<typename I , typename F > mln::transformed_image< I, F >::transformed_image (I & ima, const F & f) [inline]`

Constructor.

Definition at line 192 of file transformed_image.hh.

10.353.4 Member Function Documentation

10.353.4.1 `template<typename I , typename F > const p_transformed< typename I::domain_t, F > & mln::transformed_image< I, F >::domain () const [inline]`

Give the definition domain.

Definition at line 209 of file transformed_image.hh.

10.353.4.2 `template<typename I , typename F > mln::transformed_image< I, F >::operator transformed_image< const I, F > () const [inline]`

Const promotion via conversion.

Definition at line 239 of file transformed_image.hh.

10.353.4.3 `template<typename I , typename F > internal::morpher_lvalue_< I >::ret mln::transformed_image< I, F >::operator() (const typename I::psite & p) [inline]`

Read and "write if possible" access of pixel value at point site *p*.

Definition at line 228 of file transformed_image.hh.

10.353.4.4 `template<typename I , typename F > I::rvalue mln::transformed_image< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 217 of file transformed_image.hh.

10.354 mln::unproject_image< I, D, F > Struct Template Reference

Un-projects an image.

```
#include <unproject_image.hh>
```

Inherits image_domain_morpher< I, D, unproject_image< I, D, F > >.

Public Member Functions

- `const D & domain () const`
Give the definition domain.
- `internal::morpher_lvalue_< I >::ret operator() (const typename D::psite &p)`
*Read-write access to the image value located at point *p*.*
- `I::rvalue operator() (const typename D::psite &p) const`
*Read-only access to the image value located at point *p*.*
- `unproject_image ()`
Constructor without argument.
- `unproject_image (I &ima, const D &dom, const F &f)`
*Constructor from an image *ima*, a domain *dom*, and a function *f*.*

10.354.1 Detailed Description

template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >

Un-projects an image.

Definition at line 96 of file unproject_image.hh.

10.354.2 Constructor & Destructor Documentation

10.354.2.1 template<typename I , typename D , typename F > mln::unproject_image< I, D, F >::unproject_image () [inline]

Constructor without argument.

Definition at line 182 of file unproject_image.hh.

10.354.2.2 template<typename I , typename D , typename F > mln::unproject_image< I, D, F >::unproject_image (I & *ima*, const D & *dom*, const F & *f*) [inline]

Constructor from an image *ima*, a domain *dom*, and a function *f*.

Definition at line 188 of file unproject_image.hh.

10.354.3 Member Function Documentation

10.354.3.1 template<typename I , typename D , typename F > const D & mln::unproject_image< I, D, F >::domain () const [inline]

Give the definition domain.

Definition at line 205 of file unproject_image.hh.

10.354.3.2 template<typename I , typename D , typename F > internal::morpher_lvalue_< I >::ret mln::unproject_image< I, D, F >::operator() (const typename D::psite & *p*) [inline]

Read-write access to the image value located at point *p*.

Definition at line 225 of file unproject_image.hh.

10.354.3.3 template<typename I , typename D , typename F > I::rvalue mln::unproject_image< I, D, F >::operator() (const typename D::psite & *p*) const [inline]

Read-only access to the image value located at point *p*.

Definition at line 214 of file unproject_image.hh.

10.355 mln::util::adjacency_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits adjacency_matrix_impl_selector< V, mln::metal::equal< mln::trait::value_< V >::quant, trait::value::quant::low >::eval >.

Public Member Functions

- [adjacency_matrix](#) ()

Constructors.

- [adjacency_matrix](#) (const V &nelements)

Construct an adjacency matrix with nelements elements maximum.

10.355.1 Detailed Description

```
template<typename V = def::coord> class mln::util::adjacency_matrix< V >
```

A class of adjacency matrix. Support low and high quantification value types. In case of low quantification value type, it uses an [image2d](#) to store adjacency information. In case of high quantification value type, it uses a [util::set](#) to store the adjacency information.

Definition at line 136 of file adjacency_matrix.hh.

10.355.2 Constructor & Destructor Documentation

10.355.2.1 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ()`

Constructors.

@{

Default

Definition at line 308 of file adjacency_matrix.hh.

10.355.2.2 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix (const V & nelements)`

Construct an adjacency matrix with nelements elements maximum.

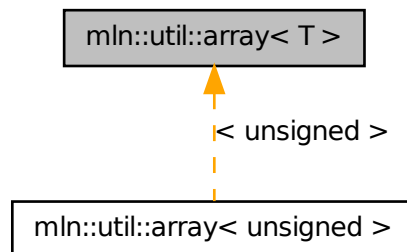
Definition at line 315 of file adjacency_matrix.hh.

10.356 mln::util::array< T > Class Template Reference

A dynamic array class.

```
#include <array.hh>
```

Inheritance diagram for `mln::util::array< T >`:



Public Types

- typedef `T` [element](#)
Element associated type.
- typedef `T` [result](#)
Returned value types.
- typedef `array_fwd_iter< T >` [fwd_eiter](#)
Iterator types
Forward iterator associated type.
- typedef `array_bkd_iter< T >` [bkd_eiter](#)
Backward iterator associated type.
- typedef [fwd_eiter](#) `eiter`
Iterator associated type.

Public Member Functions

- `array< T > & append (const T &elt)`
Add the element `elt` at the end of this array.
- `template<typename U >`
`array< T > & append (const array< U > &other)`
Add the elements of `other` at the end of this array.
- `void clear ()`

Empty the array.

- void [fill](#) (const T &value)
Fill the whole array with value value.
- bool [is_empty](#) () const
Test if the array is empty.
- ro_result [last](#) () const
Return the last element.
- mutable_result [last](#) ()
Return the last element.
- std::size_t [memory_size](#) () const
Return the size of this array in memory.
- unsigned [nelements](#) () const
Return the number of elements of the array.
- ro_result [operator\(\)](#) (unsigned i) const
*Return the *i*-th element of the array.*
- mutable_result [operator\(\)](#) (unsigned i)
*Return the *i*-th element of the array.*
- ro_result [operator\[\]](#) (unsigned i) const
*Return the *i*-th element of the array.*
- mutable_result [operator\[\]](#) (unsigned i)
*Return the *i*-th element of the array.*
- void [reserve](#) (unsigned n)
*Reserve memory for *n* elements.*
- void [resize](#) (unsigned n)
*Resize this array to *n* elements.*
- void [resize](#) (unsigned n, const T &value)
*Resize this array to *n* elements with value as value.*
- unsigned [size](#) () const
Return the number of elements of the array.
- const std::vector< T > & [std_vector](#) () const
Return the corresponding std::vector of elements.
- [array](#) ()

*Constructors**Constructor without arguments.*

- [array](#) (unsigned n)
Construct a new array and resize it to elements.
- [array](#) (unsigned n, const T &value)
Construct a new array, resize it to elements and fill it with `default_value`.

10.356.1 Detailed Description

```
template<typename T> class mln::util::array< T >
```

A dynamic array class. Elements are stored by copy. Implementation is lazy.

The parameter T is the element type, which shall not be const-qualified.

Definition at line 99 of file util/array.hh.

10.356.2 Member Typedef Documentation

10.356.2.1 `template<typename T> typedef array_bkd_iter<T> mln::util::array< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 124 of file util/array.hh.

10.356.2.2 `template<typename T> typedef fwd_eiter mln::util::array< T >::eiter`

[Iterator](#) associated type.

Definition at line 127 of file util/array.hh.

10.356.2.3 `template<typename T> typedef T mln::util::array< T >::element`

Element associated type.

Definition at line 107 of file util/array.hh.

10.356.2.4 `template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter`

[Iterator](#) types

Forward iterator associated type.

Definition at line 121 of file util/array.hh.

10.356.2.5 template<typename T> typedef T mln::util::array< T >::result

Returned value types.

Related to the [Function_v2v](#) concept.

Definition at line 112 of file util/array.hh.

10.356.3 Constructor & Destructor Documentation**10.356.3.1 template<typename T> mln::util::array< T >::array () [inline]**

Constructors

Constructor without arguments.

Definition at line 427 of file util/array.hh.

10.356.3.2 template<typename T> mln::util::array< T >::array (unsigned n) [inline]

Construct a new array and resize it to elements.

Definition at line 433 of file util/array.hh.

10.356.3.3 template<typename T> mln::util::array< T >::array (unsigned n, const T & value) [inline]

Construct a new array, resize it to elements and fill it with `default_value`.

Definition at line 440 of file util/array.hh.

10.356.4 Member Function Documentation**10.356.4.1 template<typename T> array< T > & mln::util::array< T >::append (const T & elt) [inline]**

Add the element `elt` at the end of this array.

Definition at line 472 of file util/array.hh.

Referenced by `mln::io::dicom::get_header()`, and `mln::io::plot::load()`.

10.356.4.2 template<typename T> template<typename U> array< T > & mln::util::array< T >::append (const array< U > & other) [inline]

Add the elements of `other` at the end of this array.

Definition at line 482 of file util/array.hh.

References `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::std_vector()`.

10.356.4.3 template<typename T> void mln::util::array< T >::clear () [inline]

Empty the array.

All elements contained in the array are destroyed.

Postcondition

`is_empty() == true`

Definition at line 495 of file util/array.hh.

References `mln::util::array< T >::is_empty()`.

Referenced by `mln::io::plot::load()`.

10.356.4.4 template<typename T> void mln::util::array< T >::fill (const T & value) [inline]

Fill the whole array with value `value`.

Definition at line 504 of file util/array.hh.

10.356.4.5 template<typename T> bool mln::util::array< T >::is_empty () const [inline]

Test if the array is empty.

Definition at line 578 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

Referenced by `mln::util::array< T >::append()`, `mln::util::array< T >::clear()`, `mln::make::image3d()`, and `mln::io::pnms::load()`.

10.356.4.6 template<typename T> array< T >::ro_result mln::util::array< T >::last () const [inline]

Return the last element.

Definition at line 562 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.356.4.7 template<typename T> array< T >::mutable_result mln::util::array< T >::last () [inline]

Return the last element.

Definition at line 570 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.356.4.8 template<typename T> std::size_t mln::util::array< T >::memory_size () const [inline]

Return the size of this array in memory.

Definition at line 602 of file util/array.hh.

References mln::util::array< T >::nelements().

10.356.4.9 **template<typename T > unsigned mln::util::array< T >::nelements () const** **[inline]**

Return the number of elements of the array.

Definition at line 520 of file util/array.hh.

Referenced by mln::labeling::fill_holes(), mln::make::image3d(), mln::util::array< T >::is_empty(), mln::util::array< T >::last(), mln::io::pnms::load(), mln::util::array< T >::memory_size(), mln::util::operator<<(), mln::util::array< T >::operator[](), mln::io::plot::save(), and mln::util::array< T >::size().

10.356.4.10 **template<typename T > array< T >::ro_result mln::util::array< T >::operator() (unsigned i) const** **[inline]**

Return the *i*-th element of the array.

Precondition

i < [nelements\(\)](#)

Definition at line 528 of file util/array.hh.

10.356.4.11 **template<typename T > array< T >::mutable_result mln::util::array< T >::operator() (unsigned i)** **[inline]**

Return the *i*-th element of the array.

Precondition

i < [nelements\(\)](#)

Definition at line 536 of file util/array.hh.

10.356.4.12 **template<typename T > array< T >::ro_result mln::util::array< T >::operator[] (unsigned i) const** **[inline]**

Return the *i*-th element of the array.

Precondition

i < [nelements\(\)](#)

Definition at line 544 of file util/array.hh.

References mln::util::array< T >::nelements().

10.356.4.13 `template<typename T> array< T >::mutable_result mln::util::array< T >::operator[] (unsigned i) [inline]`

Return the `i`-th element of the array.

Precondition

`i < nelements()`

Definition at line 553 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

10.356.4.14 `template<typename T> void mln::util::array< T >::reserve (unsigned n) [inline]`

Reserve memory for `n` elements.

Definition at line 448 of file `util/array.hh`.

10.356.4.15 `template<typename T> void mln::util::array< T >::resize (unsigned n) [inline]`

Resize this array to `n` elements.

Definition at line 456 of file `util/array.hh`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::io::raw::get_header()`, and `mln::io::dump::get_header()`.

10.356.4.16 `template<typename T> void mln::util::array< T >::resize (unsigned n, const T & value) [inline]`

Resize this array to `n` elements with `value` as value.

Definition at line 464 of file `util/array.hh`.

10.356.4.17 `template<typename T> unsigned mln::util::array< T >::size () const [inline]`

Return the number of elements of the array.

Added for compatibility with `fun::i2v::array`.

See also

[nelements](#)

Definition at line 512 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.356.4.18 `template<typename T> const std::vector< T > & mln::util::array< T >::std_vector () const [inline]`

Return the corresponding std::vector of elements.

Definition at line 586 of file util/array.hh.

Referenced by `mln::util::array< T >::append()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::util::operator==()`.

10.357 mln::util::branch< T > Class Template Reference

Class of generic branch.

`#include <tree.hh>`

Public Member Functions

- `tree_node< T > & apex ()`
The getter of the appex.
- `branch (tree< T > &tree, tree_node< T > &apex)`
Constructor.
- `tree< T > & util_tree ()`
The getter of the tree.

10.357.1 Detailed Description

`template<typename T> class mln::util::branch< T >`

Class of generic branch.

Definition at line 249 of file tree.hh.

10.357.2 Constructor & Destructor Documentation

10.357.2.1 `template<typename T> mln::util::branch< T >::branch (util::tree< T > & tree, util::tree_node< T > & apex) [inline]`

Constructor.

Parameters

- [in] *tree* The tree of the branch.
- [in] *apex* The apex of the branch.

Definition at line 537 of file tree.hh.

10.357.3 Member Function Documentation

10.357.3.1 `template<typename T> util::tree_node< T> & mln::util::branch< T>::apex ()` `[inline]`

The getter of the apex.

Returns

The `tree_node` apex of the current branch.

Definition at line 548 of file tree.hh.

10.357.3.2 `template<typename T> mln::util::tree< T> & mln::util::branch< T>::util_tree ()` `[inline]`

The getter of the tree.

Returns

The tree of the current branch.

Definition at line 556 of file tree.hh.

10.358 `mln::util::branch_iter< T>` Class Template Reference

Basic 2D image class.

```
#include <branch_iter.hh>
```

Public Member Functions

- unsigned `deepness` () const
Give how deep is the iterator in the branch.
- void `invalidate` ()
Invalidate the iterator.
- bool `is_valid` () const
Test the iterator validity.
- void `next` ()
Go to the next point.
- `operator util::tree_node< T> &` () const
Conversion to node.
- void `start` ()
Start an iteration.

10.358.1 Detailed Description

template<typename T> class mln::util::branch_iter< T >

Basic 2D image class. The parameter T is the type of node's data. [branch_iter](#) is used to pre-order walk a branch.

Definition at line 52 of file branch_iter.hh.

10.358.2 Member Function Documentation

**10.358.2.1 template<typename T> unsigned mln::util::branch_iter< T >::deepness () const
 [inline]**

Give how deep is the iterator in the branch.

Definition at line 119 of file branch_iter.hh.

References mln::util::branch_iter< T >::is_valid(), and mln::util::tree_node< T >::parent().

**10.358.2.2 template<typename T> void mln::util::branch_iter< T >::invalidate ()
 [inline]**

Invalidate the iterator.

Definition at line 143 of file branch_iter.hh.

Referenced by mln::util::branch_iter< T >::next().

**10.358.2.3 template<typename T> bool mln::util::branch_iter< T >::is_valid () const
 [inline]**

Test the iterator validity.

Definition at line 135 of file branch_iter.hh.

Referenced by mln::util::branch_iter< T >::deepness().

10.358.2.4 template<typename T> void mln::util::branch_iter< T >::next () [inline]

Go to the next point.

Definition at line 162 of file branch_iter.hh.

References mln::util::branch_iter< T >::invalidate().

**10.358.2.5 template<typename T> mln::util::branch_iter< T >::operator util::tree_node< T >
 & () const [inline]**

Conversion to node.

Definition at line 101 of file branch_iter.hh.

10.358.2.6 `template<typename T> void mln::util::branch_iter< T >::start () [inline]`

Start an iteration.

Definition at line 152 of file `branch_iter.hh`.

10.359 `mln::util::branch_iter_ind< T >` Class Template Reference

Basic 2D image class.

```
#include <branch_iter_ind.hh>
```

Public Member Functions

- unsigned `deepness` () const
Give how deep is the iterator in the branch.
- void `invalidate` ()
Invalidate the iterator.
- bool `is_valid` () const
Test the iterator validity.
- void `next` ()
Go to the next point.
- `operator util::tree_node< T > & ()` const
Conversion to node.
- void `start` ()
Start an iteration.

10.359.1 Detailed Description

`template<typename T> class mln::util::branch_iter_ind< T >`

Basic 2D image class. The parameter `T` is the type of node's data. `branch_iter_ind` is used to pre-order walk a branch.

Definition at line 66 of file `branch_iter_ind.hh`.

10.359.2 Member Function Documentation

10.359.2.1 `template<typename T> unsigned mln::util::branch_iter_ind< T >::deepness () const [inline]`

Give how deep is the iterator in the branch.

Definition at line 131 of file `branch_iter_ind.hh`.

References `mln::util::branch_iter_ind< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

10.359.2.2 `template<typename T> void mln::util::branch_iter_ind< T >::invalidate ()`
`[inline]`

Invalidate the iterator.

Definition at line 155 of file branch_iter_ind.hh.

Referenced by mln::util::branch_iter_ind< T >::next().

10.359.2.3 `template<typename T> bool mln::util::branch_iter_ind< T >::is_valid () const`
`[inline]`

Test the iterator validity.

Definition at line 147 of file branch_iter_ind.hh.

Referenced by mln::util::branch_iter_ind< T >::deepness().

10.359.2.4 `template<typename T> void mln::util::branch_iter_ind< T >::next () [inline]`

Go to the next point.

Definition at line 174 of file branch_iter_ind.hh.

References mln::util::branch_iter_ind< T >::invalidate().

10.359.2.5 `template<typename T> mln::util::branch_iter_ind< T >::operator util::tree_node<`
`T> & () const [inline]`

Conversion to node.

Definition at line 113 of file branch_iter_ind.hh.

10.359.2.6 `template<typename T> void mln::util::branch_iter_ind< T >::start () [inline]`

Start an iteration.

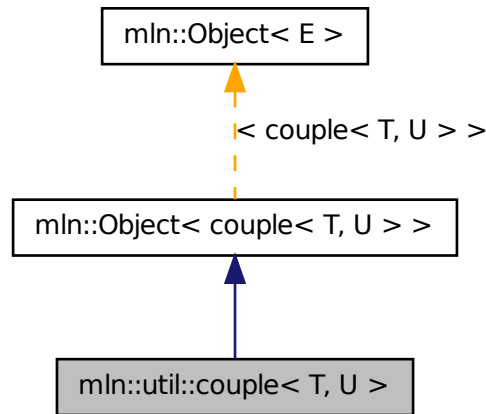
Definition at line 164 of file branch_iter_ind.hh.

10.360 mln::util::couple< T, U > Class Template Reference

Definition of a couple.

`#include <couple.hh>`

Inheritance diagram for `mln::util::couple< T, U >`:



Public Member Functions

- void `change_both` (const T &first, const U &second)
Replace both members of the couple by val.
- void `change_first` (const T &val)
Replace the first member of the couple by val.
- void `change_second` (const U &val)
Replace the second member of the couple by val.
- const T & `first` () const
Get the first member of the couple.
- const U & `second` () const
Get the second member of the couple.

10.360.1 Detailed Description

`template<typename T, typename U> class mln::util::couple< T, U >`

Definition of a couple.

Definition at line 48 of file `util/couple.hh`.

10.360.2 Member Function Documentation

10.360.2.1 `template<typename T , typename U > void mln::util::couple< T, U >::change_both (const T & first, const U & second) [inline]`

Replace both members of the couple by *val*.

Definition at line 182 of file util/couple.hh.

10.360.2.2 `template<typename T , typename U > void mln::util::couple< T, U >::change_first (const T & val) [inline]`

Replace the first member of the couple by *val*.

Definition at line 166 of file util/couple.hh.

10.360.2.3 `template<typename T , typename U > void mln::util::couple< T, U >::change_second (const U & val) [inline]`

Replace the second member of the couple by *val*.

Definition at line 174 of file util/couple.hh.

10.360.2.4 `template<typename T , typename U > const T & mln::util::couple< T, U >::first () const [inline]`

Get the first member of the couple.

Definition at line 134 of file util/couple.hh.

10.360.2.5 `template<typename T , typename U > const U & mln::util::couple< T, U >::second () const [inline]`

Get the second member of the couple.

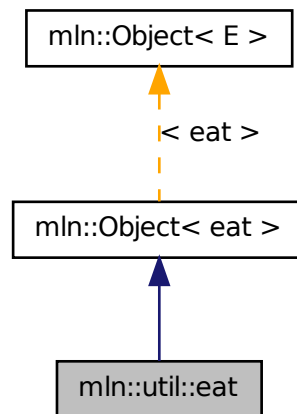
Definition at line 150 of file util/couple.hh.

10.361 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



10.361.1 Detailed Description

Eat structure.

Definition at line 46 of file eat.hh.

10.362 mln::util::edge< G > Class Template Reference

[Edge](#) of a graph G.

```
#include <edge.hh>
```

Inherits mln::util::internal::edge_impl< G >.

Public Types

- typedef [Edge](#)< void > [category](#)
Object category.
- typedef G [graph_t](#)
Graph associated type.
- typedef [edge_id_t](#) [id_t](#)
The edge type id.
- typedef [edge_id_t::value_t](#) [id_value_t](#)
The underlying type used to store edge ids.

Public Member Functions

- [edge](#) ()
Constructors.
- bool [is_valid](#) () const
Misc.
- void [invalidate](#) ()
Invalidate that vertex.
- [edge_id_t](#) [id](#) () const
Return the edge id.
- void [update_id](#) (const [edge_id_t](#) &id)
Set id_ with id;.
- [operator edge_id_t](#) () const
Conversion to the edge id.
- const G & [graph](#) () const
Return a reference to the graph holding this edge.
- void [change_graph](#) (const G &g)
Set g_ with g;.
- [vertex_id_t](#) [v_other](#) (const [vertex_id_t](#) &id_v) const
Vertex and edges oriented.
- [vertex_id_t](#) [v1](#) () const
Edge oriented.
- [vertex_id_t](#) [v2](#) () const
Return the highest vertex id adjacent to this edge.
- [size_t](#) [nmax_nbh_edges](#) () const
Return the number max of adjacent edges.
- [edge_id_t](#) [ith_nbh_edge](#) (unsigned i) const
Return the i th adjacent edge.

10.362.1 Detailed Description

`template<typename G> class mln::util::edge< G >`

[Edge](#) of a graph G.

Definition at line 69 of file edge.hh.

10.362.2 Member Typedef Documentation

10.362.2.1 `template<typename G> typedef Edge<void> mln::util::edge< G >::category`

[Object](#) category.

Definition at line 73 of file edge.hh.

10.362.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

[Graph](#) associated type.

Definition at line 82 of file edge.hh.

10.362.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The edge type id.

Definition at line 79 of file edge.hh.

10.362.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store edge ids.

Definition at line 76 of file edge.hh.

10.362.3 Constructor & Destructor Documentation

10.362.3.1 `template<typename G> mln::util::edge< G >::edge () [inline]`

Constructors.

Definition at line 227 of file edge.hh.

References `mln::util::edge< G >::invalidate()`.

10.362.4 Member Function Documentation

10.362.4.1 `template<typename G> void mln::util::edge< G >::change_graph (const G & g) [inline]`

Set `g_` with `g`;

Definition at line 290 of file edge.hh.

10.362.4.2 `template<typename G> const G & mln::util::edge< G >::graph () const [inline]`

Return a reference to the graph holding this edge.

Definition at line 282 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

10.362.4.3 template<typename G > edge_id_t mln::util::edge< G >::id () const [inline]

Return the edge id.

Definition at line 259 of file edge.hh.

Referenced by mln::util::line_graph< G >::has().

10.362.4.4 template<typename G > void mln::util::edge< G >::invalidate () [inline]

Invalidate that vertex.

Definition at line 306 of file edge.hh.

Referenced by mln::util::edge< G >::edge().

10.362.4.5 template<typename G > bool mln::util::edge< G >::is_valid () const [inline]

Misc.

Return whether is points to a known edge.

Definition at line 298 of file edge.hh.

Referenced by mln::p_edges< G, F >::has().

10.362.4.6 template<typename G > edge_id_t mln::util::edge< G >::ith_nbh_edge (unsigned i) const [inline]

Return the *i* th adjacent edge.

Definition at line 351 of file edge.hh.

10.362.4.7 template<typename G > size_t mln::util::edge< G >::nmax_nbh_edges () const [inline]

Return the number max of adjacent edges.

Definition at line 342 of file edge.hh.

10.362.4.8 template<typename G > mln::util::edge< G >::operator edge_id_t () const [inline]

Conversion to the edge id.

Definition at line 274 of file edge.hh.

10.362.4.9 template<typename G > void mln::util::edge< G >::update_id (const edge_id_t & id) [inline]

Set id_ with id;.

Definition at line 267 of file edge.hh.

10.362.4.10 template<typename G > vertex_id_t mln::util::edge< G >::v1 () const [inline]

[Edge](#) oriented.

Return the lowest vertex id adjacent to this edge.

Definition at line 324 of file edge.hh.

Referenced by mln::util::edge< G >::v_other().

10.362.4.11 template<typename G > vertex_id_t mln::util::edge< G >::v2 () const [inline]

Return the highest vertex id adjacent to this edge.

Definition at line 333 of file edge.hh.

Referenced by mln::util::edge< G >::v_other().

10.362.4.12 template<typename G > vertex_id_t mln::util::edge< G >::v_other (const vertex_id_t & id_v) const [inline]

[Vertex](#) and edges oriented.

Return the vertex id of this edge which is different from `id_v`.

Definition at line 315 of file edge.hh.

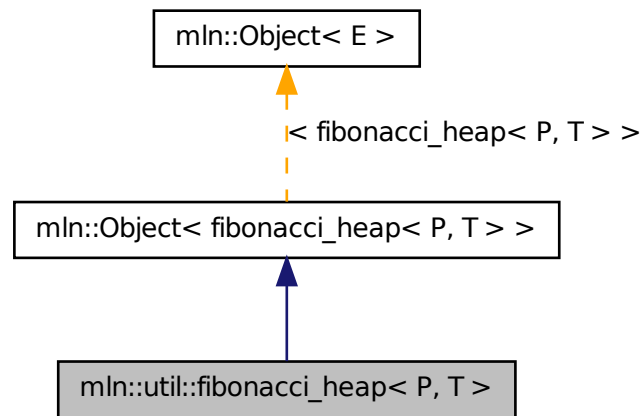
References mln::util::edge< G >::v1(), and mln::util::edge< G >::v2().

10.363 mln::util::fibonacci_heap< P, T > Class Template Reference

Fibonacci heap.

```
#include <fibonacci_heap.hh>
```

Inheritance diagram for mln::util::fibonacci_heap< P, T >:



Public Member Functions

- void `clear` ()
Clear all elements in the heap and make the heap empty.
- `fibonacci_heap` ()
Default constructor.
- `fibonacci_heap` (const `fibonacci_heap`< P, T > &`node`)
Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.
- const T & `front` () const
Return the minimum value in the heap.
- bool `is_empty` () const
Is it empty?
- bool `is_valid` () const
return false if it is empty.
- unsigned `nelements` () const
Return the number of elements.
- `fibonacci_heap`< P, T > & `operator=` (`fibonacci_heap`< P, T > &`rhs`)
Assignment operator.
- T `pop_front` ()

Return and remove the minimum value in the heap.

- void `push` (`fibonacci_heap`< P, T > &`other_heap`)

Take `other_heap`'s elements and insert them in this heap.

- void `push` (const P &`priority`, const T &`value`)

Push a new element in the heap.

10.363.1 Detailed Description

`template<typename P, typename T> class mln::util::fibonacci_heap< P, T >`

Fibonacci heap.

Definition at line 117 of file `fibonacci_heap.hh`.

10.363.2 Constructor & Destructor Documentation

10.363.2.1 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap () [inline]`

Default constructor.

Definition at line 472 of file `fibonacci_heap.hh`.

10.363.2.2 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap (const fibonacci_heap< P, T > & node) [inline]`

Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.

Definition at line 480 of file `fibonacci_heap.hh`.

10.363.3 Member Function Documentation

10.363.3.1 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::clear () [inline]`

Clear all elements in the heap and make the heap empty.

Definition at line 723 of file `fibonacci_heap.hh`.

References `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.363.3.2 `template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front () const [inline]`

Return the minimum value in the heap.

Definition at line 569 of file `fibonacci_heap.hh`.

10.363.3.3 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_empty () const [inline]`

Is it empty?

Definition at line 705 of file fibonacci_heap.hh.

Referenced by mln::util::fibonacci_heap< P, T >::pop_front(), and mln::util::fibonacci_heap< P, T >::push().

10.363.3.4 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid () const [inline]`

return false if it is empty.

Definition at line 714 of file fibonacci_heap.hh.

Referenced by mln::util::fibonacci_heap< P, T >::pop_front().

10.363.3.5 `template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements () const [inline]`

Return the number of elements.

Definition at line 745 of file fibonacci_heap.hh.

10.363.3.6 `template<typename P , typename T > fibonacci_heap< P, T > & mln::util::fibonacci_heap< P, T >::operator= (fibonacci_heap< P, T > & rhs) [inline]`

Assignment operator.

Be ware that this operator do **not** copy the data from *rhs* to this heap. It moves all elements which means that afterwards, *rhs* is is cleared and all its elements are part of this new heap.

Definition at line 754 of file fibonacci_heap.hh.

10.363.3.7 `template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front () [inline]`

Return and remove the minimum value in the heap.

Definition at line 578 of file fibonacci_heap.hh.

References mln::util::fibonacci_heap< P, T >::is_empty(), mln::util::fibonacci_heap< P, T >::is_valid(), and mln::util::fibonacci_heap< P, T >::push().

Referenced by mln::util::fibonacci_heap< P, T >::clear().

10.363.3.8 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push (const P & priority, const T & value) [inline]`

Push a new element in the heap.

See also

insert

Definition at line 508 of file fibonacci_heap.hh.

Referenced by mln::util::fibonacci_heap< P, T >::pop_front().

10.363.3.9 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push (fibonacci_heap< P, T > & other_heap) [inline]`

Take other_heap's elements and insert them in this heap.

After this call other_heap is cleared.

Definition at line 520 of file fibonacci_heap.hh.

References mln::util::fibonacci_heap< P, T >::is_empty().

10.364 mln::util::graph Class Reference

Undirected graph.

```
#include <graph.hh>
```

Inherits graph_base< graph >.

Public Types

- typedef std::set< edge_data_t > [edges_set_t](#)
A set to test the presence of a given edge.
- typedef std::vector< edge_data_t > [edges_t](#)
The type of the set of edges.
- typedef std::vector< vertex_data_t > [vertices_t](#)
The type of the set of vertices.
- typedef mln::internal::vertex_nbh_edge_fwd_iterator< [graph](#) > [vertex_nbh_edge_fwd_iter](#)
Vertex centered edge iterators.
- typedef mln::internal::vertex_nbh_vertex_fwd_iterator< [graph](#) > [vertex_nbh_vertex_fwd_iter](#)
Vertex centered vertex iterators.
- typedef mln::internal::edge_fwd_iterator< [graph](#) > [edge_fwd_iter](#)
Edge iterators.

- typedef mln::internal::edge_nbh_edge_fwd_iterator< [graph](#) > [edge_nbh_edge_fwd_iter](#)
Edge centered edge iterators.

Public Member Functions

- [graph](#) ()
- [graph](#) (unsigned nvertices)
*Construct a graph with *nvertices* vertices.*
- bool [has_v](#) (const [vertex_id_t](#) &id_v) const
*Check whether a vertex id *id_v* exists in the graph.*
- [edge_id_t](#) [v_ith_nbh_edge](#) (const [vertex_id_t](#) &id_v, unsigned i) const
*Returns the *i* th edge adjacent to the vertex *id_v*.*
- [vertex_id_t](#) [v_ith_nbh_vertex](#) (const [vertex_id_t](#) &id_v, unsigned i) const
*Returns the *i* th vertex adjacent to the vertex *id_v*.*
- size_t [v_nmax](#) () const
Return the number of vertices in the graph.
- unsigned [add_vertex](#) ()
Vertex oriented.
- std::pair< [vertex_id_t](#), [vertex_id_t](#) > [add_vertices](#) (unsigned n)
*Add *n* vertices to the graph.*
- vertex_t [vertex](#) ([vertex_id_t](#) id_v) const
*Return the vertex whose id is *v*.*
- [edge_id_t](#) [add_edge](#) (const [vertex_id_t](#) &id_v1, const [vertex_id_t](#) &id_v2)
Edge oriented.
- edge_t [edge](#) (const [edge_id_t](#) &e) const
*Return the edge whose id is *e*.*
- const std::vector< [util::ord_pair](#)< [vertex_id_t](#) > > & [edges](#) () const
Return the list of all edges.
- size_t [e_nmax](#) () const
Return the number of edges in the graph.
- bool [has_e](#) (const [edge_id_t](#) &id_e) const
*Return whether *id_e* is in the graph.*
- edge_t [edge](#) (const vertex_t &v1, const vertex_t &v2) const
Return the corresponding edge id if exists.
- [vertex_id_t](#) [v1](#) (const [edge_id_t](#) &id_e) const

Return the first vertex associated to the edge `id_e`.

- `vertex_id_t v2` (const `edge_id_t` &`id_e`) const
Return the second vertex associated to edge `id_e`.
- `edge_id_t e_ith_nbh_edge` (const `edge_id_t` &`id_e`, unsigned `i`) const
Return the `i` th edge adjacent to the edge `id_e`.
- `template<typename G2 >`
`bool is_subgraph_of` (const `G2` &`g`) const
*Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.*

10.364.1 Detailed Description

Undirected graph.

Definition at line 87 of file `mln/util/graph.hh`.

10.364.2 Member Typedef Documentation

10.364.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

[Edge](#) iterators.

Definition at line 129 of file `mln/util/graph.hh`.

10.364.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph>` `mln::util::graph::edge_nbh_edge_fwd_iter`

[Edge](#) centered edge iterators.

Definition at line 136 of file `mln/util/graph.hh`.

10.364.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A set to test the presence of a given edge.

Definition at line 102 of file `mln/util/graph.hh`.

10.364.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the set of edges.

Definition at line 100 of file `mln/util/graph.hh`.

10.364.2.5 `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph>` `mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered edge iterators.

Definition at line 115 of file `mln/util/graph.hh`.

10.364.2.6 `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph> mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered vertex iterators.

Definition at line 122 of file mln/util/graph.hh.

10.364.2.7 `typedef std::vector<vertex_data_t> mln::util::graph::vertices_t`

The type of the set of vertices.

Definition at line 97 of file mln/util/graph.hh.

10.364.3 Constructor & Destructor Documentation

10.364.3.1 `mln::util::graph::graph () [inline]`

Constructor.

Definition at line 282 of file mln/util/graph.hh.

10.364.3.2 `mln::util::graph::graph (unsigned nvertices) [inline]`

Construct a graph with `nvertices` vertices.

Definition at line 288 of file mln/util/graph.hh.

10.364.4 Member Function Documentation

10.364.4.1 `edge_id_t mln::util::graph::add_edge (const vertex_id_t & id_v1, const vertex_id_t & id_v2) [inline]`

[Edge](#) oriented.

Add an edge.

Returns

The id of the new edge if it does not exist yet; otherwise, return `mln_max(unsigned)`.

Definition at line 386 of file mln/util/graph.hh.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

10.364.4.2 `unsigned mln::util::graph::add_vertex () [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges. Add a vertex.

Returns

The id of the new vertex.

Definition at line 299 of file mln/util/graph.hh.

References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

10.364.4.3 `std::pair< vertex_id_t, vertex_id_t > mln::util::graph::add_vertices (unsigned n)` `[inline]`

Add `n` vertices to the graph.

Returns

A range of vertex ids.

Definition at line 310 of file mln/util/graph.hh.

References `v_nmax()`.

10.364.4.4 `edge_id_t mln::util::graph::e_ith_nbh_edge (const edge_id_t & id_e, unsigned i)` `const [inline]`

Return the `i` th edge adjacent to the edge `id_e`.

Definition at line 503 of file mln/util/graph.hh.

References `e_nmax()`, `has_e()`, `v1()`, `v2()`, and `v_ith_nbh_edge()`.

10.364.4.5 `size_t mln::util::graph::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 443 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

10.364.4.6 `graph::edge_t mln::util::graph::edge (const edge_id_t & e) const [inline]`

Return the edge whose id is `e`.

Definition at line 435 of file mln/util/graph.hh.

References `e_nmax()`.

Referenced by `add_edge()`.

10.364.4.7 `graph::edge_t mln::util::graph::edge (const vertex_t & v1, const vertex_t & v2)` `const [inline]`

Return the corresponding edge id if exists.

If it is not, returns an invalid edge.

Definition at line 457 of file mln/util/graph.hh.

References `has_v()`.

10.364.4.8 `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges ()
const [inline]`

Return the list of all edges.

Definition at line 428 of file mln/util/graph.hh.

10.364.4.9 `bool mln::util::graph::has_e (const edge_id_t & id_e) const [inline]`

Return whether `id_e` is in the graph.

Definition at line 450 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, `v1()`, and `v2()`.

10.364.4.10 `bool mln::util::graph::has_v (const vertex_id_t & id_v) const [inline]`

Check whether a vertex id `id_v` exists in the graph.

Definition at line 338 of file mln/util/graph.hh.

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, and `vertex()`.

10.364.4.11 `template<typename G2 > bool mln::util::graph::is_subgraph_of (const G2 & g)
const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 519 of file mln/util/graph.hh.

10.364.4.12 `vertex_id_t mln::util::graph::v1 (const edge_id_t & id_e) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 479 of file mln/util/graph.hh.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`.

10.364.4.13 `vertex_id_t mln::util::graph::v2 (const edge_id_t & id_e) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 487 of file mln/util/graph.hh.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`.

10.364.4.14 `edge_id_t mln::util::graph::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i)
const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 353 of file mln/util/graph.hh.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

10.364.4.15 `vertex_id_t mln::util::graph::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 371 of file `mln/util/graph.hh`.

References `has_v()`, and `v_ith_nbh_edge()`.

10.364.4.16 `size_t mln::util::graph::v_nmax () const [inline]`

Return the number of vertices in the graph.

Definition at line 331 of file `mln/util/graph.hh`.

Referenced by `add_vertex()`, and `add_vertices()`.

10.364.4.17 `graph::vertex_t mln::util::graph::vertex (vertex_id_t id_v) const [inline]`

Return the vertex whose id is `v`.

Definition at line 322 of file `mln/util/graph.hh`.

References `has_v()`.

10.365 mln::util::greater_point< I > Class Template Reference

A “greater than” functor comparing points w.r.t.

```
#include <greater_point.hh>
```

Public Member Functions

- bool `operator()` (const point &x, const point &y)

Is x greater than y?

10.365.1 Detailed Description

```
template<typename I> class mln::util::greater_point< I >
```

A “greater than” functor comparing points w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of points.

Definition at line 42 of file `greater_point.hh`.

10.365.2 Member Function Documentation

10.365.2.1 `template<typename I> bool mln::util::greater_point< I >::operator() (const point & x, const point & y)`

Is *x* greater than *y*?

Definition at line 74 of file greater_point.hh.

10.366 mln::util::greater_psite< I > Class Template Reference

A “greater than” functor comparing psites w.r.t.

```
#include <greater_psite.hh>
```

Public Member Functions

- `bool operator() (const psite &x, const psite &y)`

Is x greater than y?

10.366.1 Detailed Description

`template<typename I> class mln::util::greater_psite< I >`

A “greater than” functor comparing psites w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of psites.

Definition at line 42 of file greater_psite.hh.

10.366.2 Member Function Documentation

10.366.2.1 `template<typename I> bool mln::util::greater_psite< I >::operator() (const psite & x, const psite & y)`

Is *x* greater than *y*?

Definition at line 74 of file greater_psite.hh.

10.367 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```

10.367.1 Detailed Description

`template<typename T, typename R> class mln::util::head< T, R >`

Top structure of the soft heap.

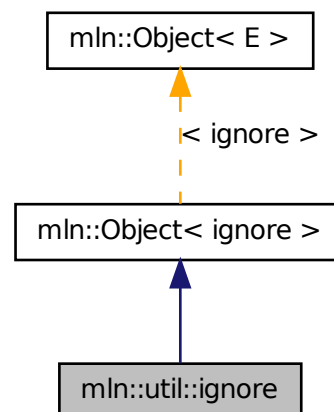
Definition at line 138 of file `soft_heap.hh`.

10.368 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```

Inheritance diagram for `mln::util::ignore`:



10.368.1 Detailed Description

Ignore structure.

Definition at line 46 of file `ignore.hh`.

10.369 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in [soft_heap](#).

```
#include <soft_heap.hh>
```

10.369.1 Detailed Description

template<typename T> struct mln::util::ilcell< T >

Element of an item list. Store the data (key) used in [soft_heap](#).

Definition at line 76 of file soft_heap.hh.

10.370 mln::util::line_graph< G > Class Template Reference

Undirected line graph of a graph of type G.

#include <line_graph.hh>

Inherits graph_base< line_graph< G > >.

Public Types

- typedef std::vector< edge_data_t > [edges_t](#)
The type of the set of edges.
- typedef std::vector< vertex_data_t > [vertices_t](#)
The type of the set of vertices.
- typedef mln::internal::edge_fwd_iterator< [line_graph](#)< G > > [edge_fwd_iter](#)
Edge iterators.
- typedef mln::internal::edge_nbh_edge_fwd_iterator< [line_graph](#)< G > > [edge_nbh_edge_fwd_iter](#)
Edge nbh edge iterators.
- typedef mln::internal::vertex_nbh_vertex_fwd_iterator< [line_graph](#)< G > > [vertex_nbh_vertex_fwd_iter](#)
Vertex nbh vertex iterators.
- typedef mln::internal::vertex_nbh_edge_fwd_iterator< [line_graph](#)< G > > [vertex_nbh_edge_fwd_iter](#)
Vertex nbh edge iterators.

Public Member Functions

- template<typename G2 >
bool [has](#) (const [util::vertex](#)< G2 > &v) const

Check whether a vertex v exists in the line graph.

- `bool has_v (const vertex_id_t &id_v) const`
Check whether a vertex id id_v exists in the line graph.
- `edge_id_t v_ith_nbh_edge (const vertex_id_t &id_v, unsigned i) const`
Returns the i th edge adjacent to the vertex id_v .
- `vertex_id_t v_ith_nbh_vertex (const vertex_id_t &id_v, unsigned i) const`
Returns the i th vertex adjacent to the vertex id_v .
- `size_t v_nmax () const`
Return the number of vertices in the graph.
- `vertex_t vertex (const vertex_id_t &id_v) const`
Vertex oriented.
- `edge_t edge (const edge_id_t &e) const`
Edge oriented.
- `size_t e_nmax () const`
Return the number of edges in the graph.
- `bool has_e (const util::edge_id_t &id_e) const`
Return whether id_e is in the line graph.
- `template<typename G2 >`
`bool has (const util::edge< G2 > &e) const`
Return whether e is in the line graph.
- `vertex_id_t v1 (const edge_id_t &id_e) const`
Return the first vertex associated to the edge id_e .
- `vertex_id_t v2 (const edge_id_t &id_e) const`
Return the second vertex associated to edge id_e .
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`
Return the i th edge adjacent to the edge id_e .
- `template<typename G2 >`
`bool is_subgraph_of (const G2 &g) const`
*Return whether this graph is a subgraph Return true if g and $*this$ have the same `graph_id`.*
- `const G & graph () const`
Return the underlying graph.

10.370.1 Detailed Description

`template<typename G> class mln::util::line_graph< G >`

Undirected line graph of a graph of type `G`.

Definition at line 82 of file line_graph.hh.

10.370.2 Member Typedef Documentation

10.370.2.1 `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G>
> mln::util::line_graph< G >::edge_fwd_iter`

[Edge](#) iterators.

Definition at line 114 of file line_graph.hh.

10.370.2.2 `template<typename G> typedef mln::internal::edge_nbh_edge_fwd_iterator<
line_graph<G> > mln::util::line_graph< G >::edge_nbh_edge_fwd_iter`

[Edge](#) nbh edge iterators.

Definition at line 123 of file line_graph.hh.

10.370.2.3 `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G
>::edges_t`

The type of the set of edges.

Definition at line 98 of file line_graph.hh.

10.370.2.4 `template<typename G> typedef mln::internal::vertex_nbh_edge_fwd_iterator<
line_graph<G> > mln::util::line_graph< G >::vertex_nbh_edge_fwd_iter`

[Vertex](#) nbh edge iterators.

Definition at line 141 of file line_graph.hh.

10.370.2.5 `template<typename G> typedef mln::internal::vertex_nbh_vertex_fwd_iterator<
line_graph<G> > mln::util::line_graph< G >::vertex_nbh_vertex_fwd_iter`

[Vertex](#) nbh vertex iterators.

Definition at line 132 of file line_graph.hh.

10.370.2.6 `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph<
G >::vertices_t`

The type of the set of vertices.

Definition at line 95 of file line_graph.hh.

10.370.3 Member Function Documentation

10.370.3.1 `template<typename G > edge_id_t mln::util::line_graph< G >::e_ith_nbh_edge (
const edge_id_t & id_e, unsigned i) const [inline]`

Return the *i* th edge adjacent to the edge *id_e*.

Definition at line 460 of file line_graph.hh.

References `mln::util::line_graph< G >::e_nmax()`, `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, and `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.370.3.2 `template<typename G > size_t mln::util::line_graph< G >::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 408 of file line_graph.hh.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::edge()`.

10.370.3.3 `template<typename G > line_graph< G >::edge_t mln::util::line_graph< G >::edge (const edge_id_t & e) const [inline]`

Edge oriented.

Return the edge whose id is *e*.

Definition at line 399 of file line_graph.hh.

References `mln::util::line_graph< G >::e_nmax()`.

10.370.3.4 `template<typename G > const G & mln::util::line_graph< G >::graph () const [inline]`

Return the underlying graph.

Definition at line 485 of file line_graph.hh.

10.370.3.5 `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has (const util::edge< G2 > & e) const [inline]`

Return whether *e* is in the line graph.

Definition at line 425 of file line_graph.hh.

References `mln::util::edge< G >::graph()`, `mln::util::line_graph< G >::has_e()`, and `mln::util::edge< G >::id()`.

10.370.3.6 `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has (const util::vertex< G2 > & v) const [inline]`

Check whether a vertex *v* exists in the line graph.

Definition at line 345 of file line_graph.hh.

References `mln::util::vertex< G >::graph()`, `mln::util::line_graph< G >::has_v()`, and `mln::util::vertex< G >::id()`.

10.370.3.7 `template<typename G> bool mln::util::line_graph< G >::has_e (const
util::edge_id_t & id_e) const [inline]`

Return whether `id_e` is in the line graph.

Definition at line 416 of file `line_graph.hh`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, `mln::util::line_graph< G >::has()`, `mln::util::line_graph< G >::v1()`, and `mln::util::line_graph< G >::v2()`.

10.370.3.8 `template<typename G> bool mln::util::line_graph< G >::has_v (const vertex_id_t
& id_v) const [inline]`

Check whether a vertex id `id_v` exists in the line graph.

Definition at line 336 of file `line_graph.hh`.

Referenced by `mln::util::line_graph< G >::has()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, `mln::util::line_graph< G >::v_ith_nbh_vertex()`, and `mln::util::line_graph< G >::vertex()`.

10.370.3.9 `template<typename G> template<typename G2> bool mln::util::line_graph< G
>::is_subgraph_of (const G2 & g) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 477 of file `line_graph.hh`.

10.370.3.10 `template<typename G> vertex_id_t mln::util::line_graph< G >::v1 (const
edge_id_t & id_e) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 433 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`.

10.370.3.11 `template<typename G> vertex_id_t mln::util::line_graph< G >::v2 (const
edge_id_t & id_e) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 442 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`.

10.370.3.12 `template<typename G> edge_id_t mln::util::line_graph< G >::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 363 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_nmax()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_ith_nbh_vertex()`.

10.370.3.13 `template<typename G> vertex_id_t mln::util::line_graph< G >::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 383 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.370.3.14 `template<typename G> size_t mln::util::line_graph< G >::v_nmax () const [inline]`

Return the number of vertices in the graph.

Definition at line 328 of file `line_graph.hh`.

Referenced by `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.370.3.15 `template<typename G> line_graph< G >::vertex_t mln::util::line_graph< G >::vertex (const vertex_id_t & id_v) const [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges.

Return the vertex whose id is `v`.

Definition at line 318 of file `line_graph.hh`.

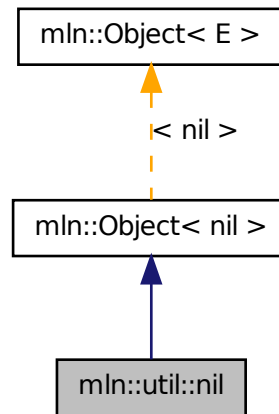
References `mln::util::line_graph< G >::has_v()`.

10.371 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```


Inheritance diagram for mln::util::nil:



10.371.1 Detailed Description

Nil structure.

Definition at line 46 of file util/nil.hh.

10.372 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

10.372.1 Detailed Description

```
template<typename T, typename R> class mln::util::node< T, R >
```

Meta-data of an element in the heap.

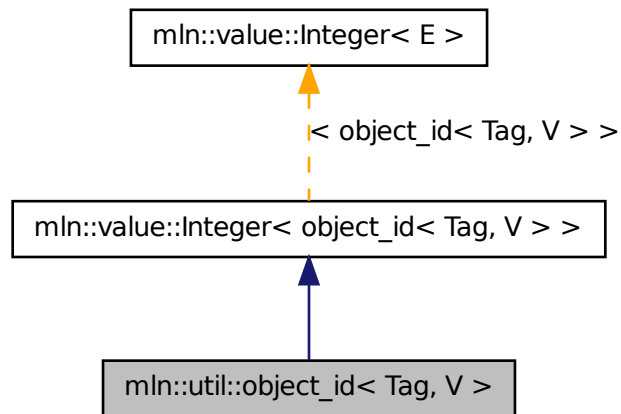
Definition at line 97 of file soft_heap.hh.

10.373 mln::util::object_id< Tag, V > Class Template Reference

Base class of an object id.

```
#include <object_id.hh>
```

Inheritance diagram for `mln::util::object_id< Tag, V >`:



Public Types

- typedef V [value_t](#)

The underlying type id.

Public Member Functions

- [object_id\(\)](#)

Constructors.

10.373.1 Detailed Description

template<typename Tag, typename V> class mln::util::object_id< Tag, V >

Base class of an object id.

Template Parameters

Tag the tag type

Equiv the equivalent value.

Definition at line 66 of file `object_id.hh`.

10.373.2 Member Typedef Documentation

10.373.2.1 `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

Definition at line 70 of file object_id.hh.

10.373.3 Constructor & Destructor Documentation

10.373.3.1 `template<typename Tag , typename V > mln::util::object_id< Tag, V >::object_id () [inline]`

Constructors.

Definition at line 120 of file object_id.hh.

10.374 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type T : *lhs R rhs*.

```
#include <ord.hh>
```

10.374.1 Detailed Description

`template<typename T> struct mln::util::ord< T >`

Function-object that defines an ordering between objects with type T : *lhs R rhs*. Its meaning is "lhs less-than rhs."

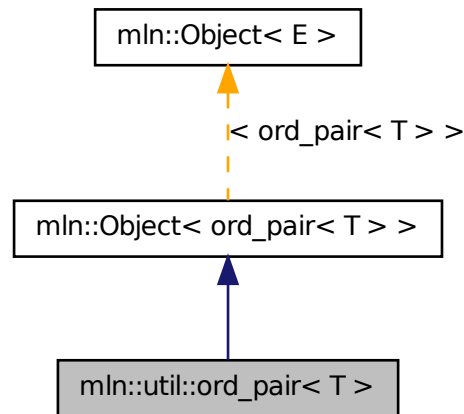
Definition at line 48 of file util/ord.hh.

10.375 mln::util::ord_pair< T > Struct Template Reference

Ordered pair structure s.a.

```
#include <ord_pair.hh>
```

Inheritance diagram for `mln::util::ord_pair< T >`:



Public Member Functions

- void `change_both` (const T &first, const T &second)
Replace both members of the pair by val, while keeping the relative order.
- void `change_first` (const T &val)
Replace the first member of the pair by val, while keeping the relative order.
- void `change_second` (const T &val)
Replace the second member of the pair by val, while keeping the relative order.
- const T & `first` () const
Get the first (lowest) member of the pair.
- const T & `second` () const
Get the second (highest) member of the pair.

10.375.1 Detailed Description

`template<typename T> struct mln::util::ord_pair< T >`

Ordered pair structure s.a. `this->first <= this->second`; ordered pairs are partially ordered using lexicographical ordering.

Definition at line 50 of file `ord_pair.hh`.

10.375.2 Member Function Documentation

10.375.2.1 `template<typename T> void mln::util::ord_pair< T >::change_both (const T & first, const T & second) [inline]`

Replace both members of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= *second_* (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 211 of file ord_pair.hh.

References [mln::util::ord_strict\(\)](#), and [mln::util::ord_weak\(\)](#).

10.375.2.2 `template<typename T> void mln::util::ord_pair< T >::change_first (const T & val) [inline]`

Replace the first member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= *second_* (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 181 of file ord_pair.hh.

References [mln::util::ord_strict\(\)](#), and [mln::util::ord_weak\(\)](#).

10.375.2.3 `template<typename T> void mln::util::ord_pair< T >::change_second (const T & val) [inline]`

Replace the second member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= *second_* (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 196 of file ord_pair.hh.

References [mln::util::ord_strict\(\)](#), and [mln::util::ord_weak\(\)](#).

10.375.2.4 `template<typename T> const T & mln::util::ord_pair< T >::first () const [inline]`

Get the first (lowest) member of the pair.

Definition at line 149 of file ord_pair.hh.

10.375.2.5 `template<typename T> const T & mln::util::ord_pair< T >::second () const [inline]`

Get the second (highest) member of the pair.

Definition at line 165 of file ord_pair.hh.

10.376 mln::util::pix< I > Struct Template Reference

Structure pix.

```
#include <pix.hh>
```

Public Types

- typedef I::psite [psite](#)
Point_Site associated type.
- typedef I::value [value](#)
Value associated type.

Public Member Functions

- const I & [ima](#) () const
The getter of the image associate to pix structure.
- const I::psite & [p](#) () const
The getter of psite associate to pix structure.
- [pix](#) (const [Image](#)< I > &ima, const typename I::psite &p)
Constructor.
- I::rvalue [v](#) () const
The getter of value associate to pix structure.

10.376.1 Detailed Description

template<typename I> struct mln::util::pix< I >

Structure pix.

Definition at line 69 of file util/pix.hh.

10.376.2 Member Typedef Documentation

10.376.2.1 template<typename I> typedef I ::psite mln::util::pix< I >::psite

Point_Site associated type.

Definition at line 73 of file util/pix.hh.

10.376.2.2 template<typename I> typedef I ::value mln::util::pix< I >::value

[Value](#) associated type.

Definition at line 76 of file util/pix.hh.

10.376.3 Constructor & Destructor Documentation

10.376.3.1 `template<typename I> mln::util::pix< I >::pix (const Image< I > & ima, const typename I::psite & p) [inline]`

Constructor.

Parameters

[in] *ima* The image.

[in] *p* The p_site.

Definition at line 121 of file util/pix.hh.

10.376.4 Member Function Documentation

10.376.4.1 `template<typename I> const I & mln::util::pix< I >::ima () const [inline]`

The getter of the image associate to pix structure.

Returns

The image ima_.

Definition at line 131 of file util/pix.hh.

10.376.4.2 `template<typename I> const I::psite & mln::util::pix< I >::p () const [inline]`

The getter of psite associate to pix structure.

Returns

The psite p_.

Definition at line 140 of file util/pix.hh.

10.376.4.3 `template<typename I> I::rvalue mln::util::pix< I >::v () const [inline]`

The getter of value associate to pix structure.

Returns

The value of pix.

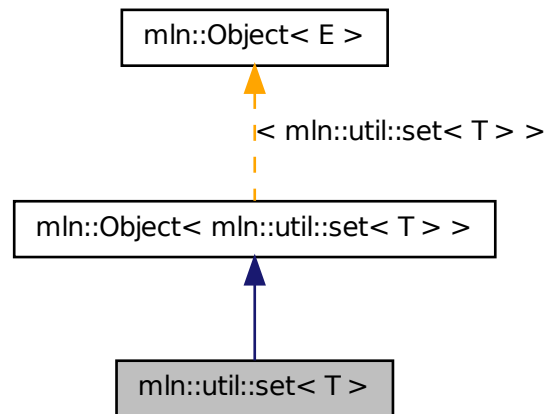
Definition at line 150 of file util/pix.hh.

10.377 mln::util::set< T > Class Template Reference

An "efficient" mathematical set class.

```
#include <set.hh>
```

Inheritance diagram for `mln::util::set< T >`:



Public Types

- typedef `set_bkd_iter< T >` `bkd_eiter`
Backward iterator associated type.
- typedef `fwd_eiter eiter`
Iterator associated type.
- typedef `T element`
Element associated type.
- typedef `set_fwd_iter< T >` `fwd_eiter`
Forward iterator associated type.

Public Member Functions

- void `clear ()`
Empty the set.
- const `T first_element () const`
Return the first element of the set.
- bool `has (const T &elt) const`
Test if the object `elt` belongs to the set.
- `set< T > & insert (const T &elt)`

Insert an element `elt` into the set.

- `template<typename U >`
`set< T > & insert (const set< U > &other)`

Insert the elements of `other` into the set.

- `bool is_empty () const`

Test if the set is empty.

- `const T last_element () const`

Return the last element of the set.

- `std::size_t memory_size () const`

Return the size of this set in memory.

- `unsigned nelements () const`

Return the number of elements of the set.

- `const T & operator[] (unsigned i) const`

*Return the *i*-th element of the set.*

- `set< T > & remove (const T &elt)`

Remove an element `elt` into the set.

- `set ()`

Constructor without arguments.

- `const std::vector< T > & std_vector () const`

Give access to the set elements.

10.377.1 Detailed Description

template<typename T> class mln::util::set< T >

An "efficient" mathematical set class. This set class is designed to store a mathematical set and to present it to the user as a linear array (`std::vector`).

Elements are stored by copy. Implementation is lazy.

The set has two states: frozen or not. There is an automatic switch of state when the user modifies its contents (insert, remove, or clear) or access to its contents (`op[i]`).

The parameter `T` is the element type, which shall not be const-qualified.

The unicity of set elements is handled by the [mln::util::ord](#) mechanism.

See also

[mln::util::ord](#)

Definition at line 81 of file `util/set.hh`.

10.377.2 Member Typedef Documentation

10.377.2.1 `template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 93 of file util/set.hh.

10.377.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

[Iterator](#) associated type.

Definition at line 96 of file util/set.hh.

10.377.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

Definition at line 86 of file util/set.hh.

10.377.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

Definition at line 90 of file util/set.hh.

10.377.3 Constructor & Destructor Documentation

10.377.3.1 `template<typename T> mln::util::set< T >::set () [inline]`

Constructor without arguments.

Definition at line 348 of file util/set.hh.

10.377.4 Member Function Documentation

10.377.4.1 `template<typename T> void mln::util::set< T >::clear () [inline]`

Empty the set.

All elements contained in the set are destroyed so the set is emptied.

Postcondition

`is_empty() == true`

Definition at line 390 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

10.377.4.2 `template<typename T> const T mln::util::set< T >::first_element () const [inline]`

Return the first element of the set.

Precondition

not `is_empty()`

Definition at line 427 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

**10.377.4.3 template<typename T > bool mln::util::set< T >::has (const T & elt) const
[inline]**

Test if the object `elt` belongs to the set.

Parameters

[in] *elt* A possible element of the set.

Returns

True is `elt` is in the set.

Definition at line 445 of file util/set.hh.

**10.377.4.4 template<typename T > set< T > & mln::util::set< T >::insert (const T & elt)
[inline]**

Insert an element `elt` into the set.

Parameters

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after insertion.

Definition at line 356 of file util/set.hh.

Referenced by `mln::p_key< K, P >::change_keys()`.

**10.377.4.5 template<typename T > template<typename U > set< T > & mln::util::set< T
>::insert (const set< U > & other) [inline]**

Insert the elements of `other` into the set.

Parameters

[in] *other* The set containing the elements to be inserted.

Returns

The set itself after insertion.

Definition at line 367 of file util/set.hh.

References `mln::util::set< T >::is_empty()`, and `mln::util::set< T >::std_vector()`.

10.377.4.6 template<typename T> bool mln::util::set< T>::is_empty () const [inline]

Test if the set is empty.

Definition at line 453 of file util/set.hh.

References mln::util::set< T>::nelements().

Referenced by mln::util::set< T>::clear(), mln::util::set< T>::first_element(), mln::util::set< T>::insert(), and mln::util::set< T>::last_element().

**10.377.4.7 template<typename T> const T mln::util::set< T>::last_element () const
 [inline]**

Return the last element of the set.

Precondition

not [is_empty\(\)](#)

Definition at line 436 of file util/set.hh.

References mln::util::set< T>::is_empty().

**10.377.4.8 template<typename T> std::size_t mln::util::set< T>::memory_size () const
 [inline]**

Return the size of this set in memory.

Definition at line 494 of file util/set.hh.

References mln::util::set< T>::nelements().

**10.377.4.9 template<typename T> unsigned mln::util::set< T>::nelements () const
 [inline]**

Return the number of elements of the set.

Definition at line 409 of file util/set.hh.

Referenced by mln::util::set< T>::is_empty(), mln::util::set< T>::memory_size(), and mln::util::set< T>::operator[]().

**10.377.4.10 template<typename T> const T & mln::util::set< T>::operator[] (unsigned i)
 const [inline]**

Return the i-th element of the set.

Parameters

[in] *i* Index of the element to retrieve.

Precondition

i < [nelements\(\)](#)

The element is returned by reference and is constant.

Definition at line 417 of file util/set.hh.

References mln::util::set< T >::nelements().

10.377.4.11 `template<typename T> set< T > & mln::util::set< T >::remove (const T & elt)` `[inline]`

Remove an element `elt` into the set.

Parameters

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after suppression.

Definition at line 380 of file util/set.hh.

10.377.4.12 `template<typename T> const std::vector< T > & mln::util::set< T >::std_vector () const` `[inline]`

Give access to the set elements.

The complexity of this method is O(1).

Postcondition

The set is frozen.

Returns

An array (std::vector) of elements.

Definition at line 461 of file util/set.hh.

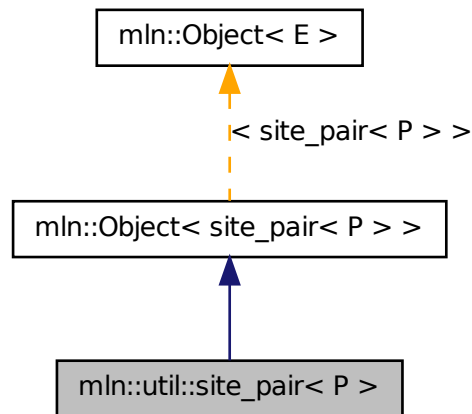
Referenced by mln::util::set< T >::insert().

10.378 mln::util::site_pair< P > Class Template Reference

A pair of sites.

```
#include <site_pair.hh>
```

Inheritance diagram for `mln::util::site_pair< P >`:



Public Member Functions

- `const P & first () const`
Return the first site.
- `const util::ord_pair< P > & pair () const`
Return the underlying pair.
- `const P & second () const`
Return the second site.

10.378.1 Detailed Description

`template<typename P> class mln::util::site_pair< P >`

A pair of sites. It can be used as site.

Definition at line 52 of file `site_pair.hh`.

10.378.2 Member Function Documentation

10.378.2.1 `template<typename P > const P & mln::util::site_pair< P >::first () const`
`[inline]`

Return the first site.

Definition at line 142 of file `site_pair.hh`.

10.378.2.2 `template<typename P> const util::ord_pair< P> & mln::util::site_pair< P>::pair () const [inline]`

Return the underlying pair.

Definition at line 158 of file site_pair.hh.

10.378.2.3 `template<typename P> const P & mln::util::site_pair< P>::second () const [inline]`

Return the second site.

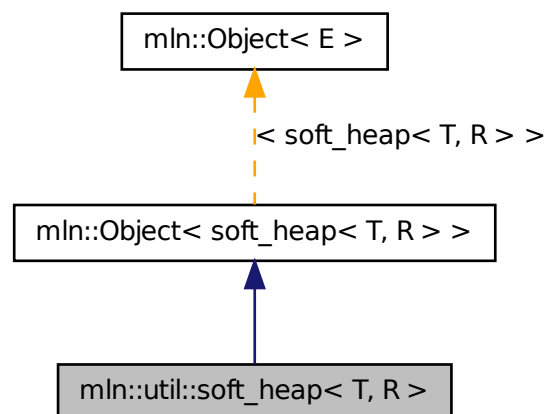
Definition at line 150 of file site_pair.hh.

10.379 mln::util::soft_heap< T, R > Class Template Reference

Soft heap.

```
#include <soft_heap.hh>
```

Inheritance diagram for mln::util::soft_heap< T, R >:



Public Types

- typedef T [element](#)
Element associated type.

Public Member Functions

- void [clear](#) ()

Clear the heap.

- bool `is_empty ()` const

Return true if there is at least one element.

- bool `is_valid ()` const

Return true if there is at least one element.

- int `nelements ()` const

Return the number of element in the heap.

- T `pop_front ()`

Returns the element with the lowest priority and remove it from the heap.

- void `push (soft_heap< T, R > &sh)`

Merge sh with this heap.

- void `push (const T &element)`

Add a new element element.

- `soft_heap` (unsigned r=20)

Default constructor.

- `~soft_heap ()`

Destructor.

10.379.1 Detailed Description

template<typename T, typename R> class mln::util::soft_heap< T, R >

Soft heap. T key, the data to store in the heap. For instance a point 2d. R rank, for instance int_u8

Definition at line 178 of file soft_heap.hh.

10.379.2 Member Typedef Documentation

10.379.2.1 template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element

Element associated type.

Definition at line 185 of file soft_heap.hh.

10.379.3 Constructor & Destructor Documentation

10.379.3.1 template<typename T , typename R > mln::util::soft_heap< T, R >::soft_heap (unsigned r = 20) [inline]

Default constructor.

A corruption threshold τ can be specified. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced.

Definition at line 619 of file soft_heap.hh.

10.379.3.2 `template<typename T , typename R > mln::util::soft_heap< T, R >::~~soft_heap ()`
`[inline]`

Destructor.

Definition at line 631 of file soft_heap.hh.

10.379.4 Member Function Documentation

10.379.4.1 `template<typename T , typename R > void mln::util::soft_heap< T, R >::clear ()`
`[inline]`

Clear the heap.

Definition at line 771 of file soft_heap.hh.

10.379.4.2 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_empty () const` `[inline]`

Return true if there is at least one element.

Definition at line 753 of file soft_heap.hh.

10.379.4.3 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_valid () const` `[inline]`

Return true if there is at least one element.

Definition at line 744 of file soft_heap.hh.

Referenced by mln::util::soft_heap< T, R >::pop_front().

10.379.4.4 `template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements () const` `[inline]`

Return the number of element in the heap.

Definition at line 762 of file soft_heap.hh.

Referenced by mln::util::soft_heap< T, R >::push().

10.379.4.5 `template<typename T , typename R > T mln::util::soft_heap< T, R >::pop_front ()`
`[inline]`

Returns the element with the lowest priority and remove it from the heap.

Definition at line 675 of file soft_heap.hh.

References mln::util::soft_heap< T, R >::is_valid().

10.379.4.6 `template<typename T , typename R > void mln::util::soft_heap< T, R >::push (const T & element) [inline]`

Add a new element `element`.

Definition at line 646 of file `soft_heap.hh`.

10.379.4.7 `template<typename T , typename R > void mln::util::soft_heap< T, R >::push (soft_heap< T, R > & sh) [inline]`

Merge `sh` with this heap.

Be ware that after this call, `sh` will be empty. This heap will hold the elements which were part of `sh`.

Definition at line 658 of file `soft_heap.hh`.

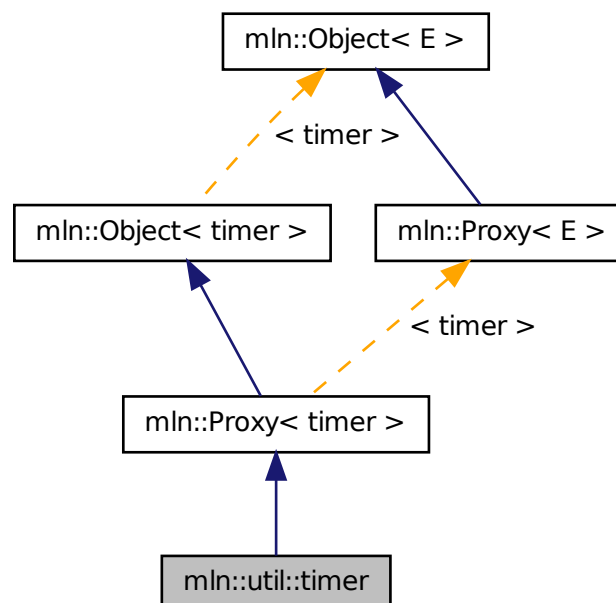
References `mln::util::soft_heap< T, R >::nelements()`.

10.380 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for `mln::util::timer`:



10.380.1 Detailed Description

Timer structure.

Definition at line 45 of file mln/util/timer.hh.

10.381 mln::util::tracked_ptr< T > Struct Template Reference

Smart pointer for shared data with tracking.

```
#include <tracked_ptr.hh>
```

Public Member Functions

- `operator bool () const`
Coercion towards Boolean (for arithmetical tests).
- `bool operator! () const`
Negation (for arithmetical tests).
- `T * operator-> ()`
Mimics the behavior of op-> for a pointer in the mutable case.
- `const T * operator-> () const`
Mimics the behavior of op-> for a pointer in the const case.
- `tracked_ptr< T > & operator= (T *ptr)`
Assignment.
- `tracked_ptr< T > & operator= (const tracked_ptr< T > &rhs)`
Assignment.
- `~tracked_ptr ()`
Destructor.
- `tracked_ptr ()`
Constructors.
- `tracked_ptr (const tracked_ptr< T > &rhs)`
Copy constructor.

10.381.1 Detailed Description

```
template<typename T> struct mln::util::tracked_ptr< T >
```

Smart pointer for shared data with tracking.

Definition at line 52 of file tracked_ptr.hh.

10.381.2 Constructor & Destructor Documentation

10.381.2.1 `template<typename T> mln::util::tracked_ptr< T>::tracked_ptr () [inline]`

Constructors.

Definition at line 140 of file tracked_ptr.hh.

10.381.2.2 `template<typename T> mln::util::tracked_ptr< T>::tracked_ptr (const tracked_ptr< T> & rhs) [inline]`

Copy constructor.

Definition at line 164 of file tracked_ptr.hh.

10.381.2.3 `template<typename T> mln::util::tracked_ptr< T>::~~tracked_ptr () [inline]`

Destructor.

Definition at line 216 of file tracked_ptr.hh.

10.381.3 Member Function Documentation

10.381.3.1 `template<typename T> mln::util::tracked_ptr< T>::operator bool () const [inline]`

Coercion towards Boolean (for arithmetical tests).

Definition at line 106 of file tracked_ptr.hh.

10.381.3.2 `template<typename T> bool mln::util::tracked_ptr< T>::operator! () const [inline]`

Negation (for arithmetical tests).

Definition at line 114 of file tracked_ptr.hh.

10.381.3.3 `template<typename T> T * mln::util::tracked_ptr< T>::operator-> () [inline]`

Mimics the behavior of `op->` for a pointer in the mutable case.

Invariant

Pointer proxy exists.

Definition at line 131 of file tracked_ptr.hh.

10.381.3.4 `template<typename T> const T * mln::util::tracked_ptr< T>::operator-> () const [inline]`

Mimics the behavior of `op->` for a pointer in the const case.

Invariant

Pointer proxy exists.

Definition at line 122 of file tracked_ptr.hh.

10.381.3.5 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (T * ptr) [inline]`

Assignment.

Definition at line 195 of file tracked_ptr.hh.

10.381.3.6 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (const tracked_ptr< T > & rhs) [inline]`

Assignment.

Definition at line 176 of file tracked_ptr.hh.

10.382 mln::util::tree< T > Class Template Reference

Class of generic tree.

```
#include <tree.hh>
```

Public Member Functions

- void [add_tree_down](#) (T &elt)
Bind a new tree downner the current.
- void [add_tree_up](#) (T &elt)
Bind a new tree upper the current.
- bool [check_consistency](#) ()
Check the consistency of the tree.
- [branch](#)< T > [main_branch](#) ()
Convert the tree into brach.
- [tree_node](#)< T > * [root](#) ()
The getter of the root.
- [tree](#) ()
Constructor.
- [tree](#) ([tree_node](#)< T > *root)
Constructor.

10.382.1 Detailed Description

template<typename T> class mln::util::tree< T >

Class of generic tree.

Definition at line 187 of file tree.hh.

10.382.2 Constructor & Destructor Documentation

10.382.2.1 template<typename T> mln::util::tree< T >::tree () [inline]

Constructor.

Definition at line 285 of file tree.hh.

10.382.2.2 template<typename T> mln::util::tree< T >::tree (tree_node< T > * root) [inline]

Constructor.

Parameters

[in] *root* The root of the tree.

Definition at line 292 of file tree.hh.

10.382.3 Member Function Documentation

10.382.3.1 template<typename T> void mln::util::tree< T >::add_tree_down (T & elt) [inline]

Bind a new tree downer the current.

Parameters

[in] *elt* The new value of the new [tree_node](#) of the new tree add downer the current.

Definition at line 328 of file tree.hh.

10.382.3.2 template<typename T> void mln::util::tree< T >::add_tree_up (T & elt) [inline]

Bind a new tree upper the current.

Parameters

[in] *elt* The new value of the new [tree_node](#) of the new tree add upper the current.

Definition at line 317 of file tree.hh.

References `mln::util::tree_node< T >::children()`.

10.382.3.3 `template<typename T> bool mln::util::tree< T >::check_consistency ()` `[inline]`

Check the consistency of the tree.

Returns

true if no error, else false.

Definition at line 338 of file tree.hh.

References mln::util::tree< T >::root().

10.382.3.4 `template<typename T> branch< T > mln::util::tree< T >::main_branch ()` `[inline]`

Convert the tree into brach.

Returns

The root's [tree_node](#) of the the current tree.

Definition at line 309 of file tree.hh.

References mln::util::tree< T >::root().

10.382.3.5 `template<typename T> tree_node< T > * mln::util::tree< T >::root ()` `[inline]`

The getter of the root.

Returns

The root's [tree_node](#) of the the current tree.

Definition at line 301 of file tree.hh.

Referenced by mln::util::tree< T >::check_consistency(), mln::util::display_tree(), mln::util::tree< T >::main_branch(), and mln::util::tree_to_fast().

10.383 mln::util::tree_node< T > Class Template Reference

Class of generic [tree_node](#) for tree.

```
#include <tree.hh>
```

Public Member Functions

- [tree_node](#)< T > * [add_child](#) (T elt)
Create a [tree_node](#) with `elt` which become the child of the current [tree_node](#).
- [tree_node](#)< T > * [add_child](#) ([tree_node](#)< T > *[tree_node](#))

Bind `tree_node` to the current `tree_node` and become its child.

- `bool check_consistency ()`
Check the consistency of the `tree_node`.
- `children_t & children ()`
The getter of the children.
- `const children_t & children () const`
The getter of the children.
- `tree_node< T > * delete_tree_node ()`
Delete the current `tree_node`.
- `T & elt ()`
The getter of the element.
- `const T & elt () const`
The const getter of the element.
- `tree_node< T > * parent ()`
The getter of the parent.
- `void print (std::ostream &ostr, int level=0)`
Print on `ostr` the arborescence with the current `tree_node` as root.
- `tree_node< T > * search (T &elt)`
Search the `tree_node` with value `elt` in the arborescence of the current `tree_node`.
- `int search_rec (tree_node< T > **res, T &elt)`
The using method for method search.
- `void set_parent (tree_node< T > *parent)`
Bind `tree_node` to the current `tree_node` and become its parent.
- `tree_node ()`
Constructor.
- `tree_node (T elt)`
Constructor.

10.383.1 Detailed Description

`template<typename T> class mln::util::tree_node< T >`

Class of generic `tree_node` for tree.

Definition at line 58 of file `tree.hh`.

10.383.2 Constructor & Destructor Documentation

10.383.2.1 `template<typename T> mln::util::tree_node< T >::tree_node () [inline]`

Constructor.

Definition at line 345 of file tree.hh.

10.383.2.2 `template<typename T> mln::util::tree_node< T >::tree_node (T elt) [inline]`

Constructor.

Parameters

[in] *elt* The element of [tree_node](#).

Definition at line 352 of file tree.hh.

10.383.3 Member Function Documentation

10.383.3.1 `template<typename T> tree_node< T > * mln::util::tree_node< T >::add_child (T elt) [inline]`

Create a [tree_node](#) with *elt* which become the child of the current [tree_node](#).

Parameters

[in] *elt* The element of the new child to add.

Returns

The new [tree_node](#) created.

Definition at line 394 of file tree.hh.

10.383.3.2 `template<typename T> tree_node< T > * mln::util::tree_node< T >::add_child (tree_node< T > * tree_node) [inline]`

Bind [tree_node](#) to the current [tree_node](#) and become its child.

Parameters

[in] *tree_node* The new child [tree_node](#).

Returns

The child [tree_node](#).

Definition at line 407 of file tree.hh.

References [mln::util::tree_node< T >::children\(\)](#), and [mln::util::tree_node< T >::parent\(\)](#).

10.383.3.3 `template<typename T> bool mln::util::tree_node< T>::check_consistency ()`
`[inline]`

Check the consistency of the [tree_node](#).

Returns

true if no error, else false.

Definition at line 519 of file tree.hh.

10.383.3.4 `template<typename T> const std::vector< tree_node< T> * > &`
`mln::util::tree_node< T>::children () const [inline]`

The getter of the children.

Returns

The children of the [tree_node](#) in const.

Definition at line 386 of file tree.hh.

10.383.3.5 `template<typename T> std::vector< tree_node< T> * > & mln::util::tree_node< T`
`>::children () [inline]`

The getter of the children.

Returns

The children of the [tree_node](#).

Definition at line 378 of file tree.hh.

Referenced by `mln::util::tree_node< T>::add_child()`, and `mln::util::tree< T>::add_tree_up()`.

10.383.3.6 `template<typename T> tree_node< T> * mln::util::tree_node< T`
`>::delete_tree_node () [inline]`

Delete the current [tree_node](#).

Definition at line 427 of file tree.hh.

10.383.3.7 `template<typename T> const T & mln::util::tree_node< T>::elt () const`
`[inline]`

The const getter of the element.

Returns

The element of the [tree_node](#) in const.

Definition at line 361 of file tree.hh.

10.383.3.8 `template<typename T> T & mln::util::tree_node< T >::elt () [inline]`

The getter of the element.

Returns

The element of the [tree_node](#).

Definition at line 369 of file tree.hh.

Referenced by `mln::util::tree_node< T >::print()`.

10.383.3.9 `template<typename T> tree_node< T > * mln::util::tree_node< T >::parent () [inline]`

The getter of the parent.

Returns

The parent of the [tree_node](#).

Definition at line 477 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter_ind< T >::deepness()`, and `mln::util::branch_iter< T >::deepness()`.

10.383.3.10 `template<typename T> void mln::util::tree_node< T >::print (std::ostream & ostr, int level = 0) [inline]`

Print on `ostr` the arborescence with the current [tree_node](#) as root.

Parameters

[in] *ostr* The output stream.

[in] *level* The deep level

Definition at line 449 of file tree.hh.

References `mln::util::tree_node< T >::elt()`.

10.383.3.11 `template<typename T> tree_node< T > * mln::util::tree_node< T >::search (T & elt) [inline]`

Search the [tree_node](#) with value `elt` in the arborescence of the current [tree_node](#).

Parameters

[in] *elt* The value of the searched [tree_node](#).

Returns

If not found 0 else the [tree_node](#) with `elt` value.

Definition at line 507 of file tree.hh.

References `mln::util::tree_node< T >::search_rec()`.

10.383.3.12 `template<typename T> int mln::util::tree_node< T >::search_rec (tree_node< T > ** res, T & elt) [inline]`

The using method for method search.

Definition at line 485 of file tree.hh.

Referenced by `mln::util::tree_node< T >::search()`.

10.383.3.13 `template<typename T> void mln::util::tree_node< T >::set_parent (tree_node< T > * parent) [inline]`

Bind `tree_node` to the current `tree_node` and become its parent.

Parameters

[in] *parent* The new parent `tree_node`.

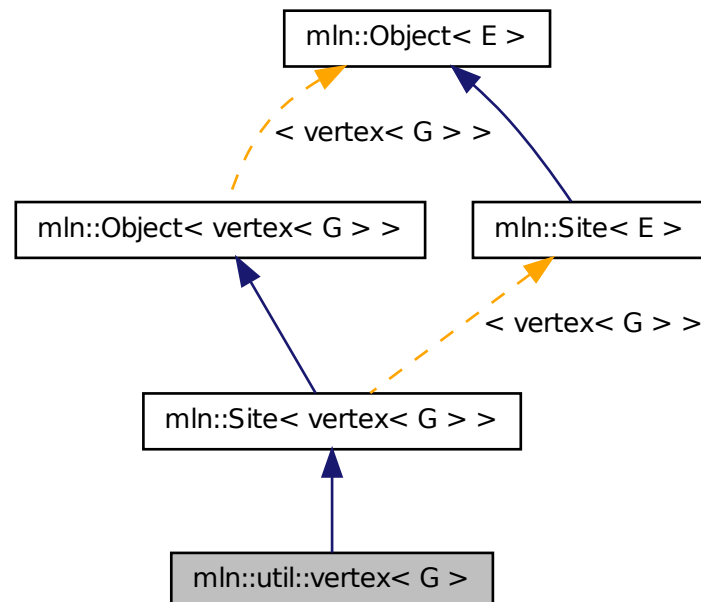
Definition at line 467 of file tree.hh.

10.384 mln::util::vertex< G > Class Template Reference

`Vertex` of a graph `G`.

```
#include <vertex.hh>
```

Inheritance diagram for `mln::util::vertex< G >`:



Public Types

- typedef [Vertex](#)< void > [Category](#)
Object category.
- typedef G [graph_t](#)
Graph associated type.
- typedef [vertex_id_t](#) [id_t](#)
The vertex type id.
- typedef [vertex_id_t::value_t](#) [id_value_t](#)
The underlying type used to store vertex ids.

Public Member Functions

- void [change_graph](#) (const G &g)
Change the parent graph of that vertex.
- [edge](#)< G > [edge_with](#) (const [vertex](#)< G > &v_id) const
Returns true if this vertex has an edge with the given vertex.
- const G & [graph](#) () const
Returns the graph pointer this vertex belongs to.
- const [vertex_id_t](#) & [id](#) () const
Returns the vertex id.
- void [invalidate](#) ()
Invalidate that vertex.
- bool [is_valid](#) () const
Check whether the vertex is still part of the graph.
- [edge_id_t](#) [ith_nbh_edge](#) (unsigned i) const
Returns the ith edge starting from this vertex.
- [vertex_id_t](#) [ith_nbh_vertex](#) (unsigned i) const
Returns the ith vertex adjacent to this vertex.
- unsigned [nmax_nbh_edges](#) () const
Returns the number max of edges starting from this vertex.
- unsigned [nmax_nbh_vertices](#) () const
Returns the number max of vertices adjacent to this vertex.
- operator [vertex_id_t](#) () const
Conversion to the vertex id.

- [vertex_id_t other](#) (const [edge_id_t](#) &id_e) const
Returns the other vertex located on edge `id_e`.
- void [update_id](#) (const [vertex_id_t](#) &id)
Update the vertex id.
- [vertex](#) ()
Constructors.

10.384.1 Detailed Description

template<typename G> class mln::util::vertex< G >

[Vertex](#) of a graph G.

Definition at line 71 of file vertex.hh.

10.384.2 Member Typedef Documentation

10.384.2.1 template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category

[Object](#) category.

Definition at line 77 of file vertex.hh.

10.384.2.2 template<typename G> typedef G mln::util::vertex< G >::graph_t

[Graph](#) associated type.

Definition at line 86 of file vertex.hh.

10.384.2.3 template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t

The vertex type id.

Definition at line 83 of file vertex.hh.

10.384.2.4 template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t

The underlying type used to store vertex ids.

Definition at line 80 of file vertex.hh.

10.384.3 Constructor & Destructor Documentation

10.384.3.1 template<typename G> mln::util::vertex< G >::vertex () [inline]

Constructors.

Definition at line 226 of file vertex.hh.

References mln::util::vertex< G >::invalidate().

10.384.4 Member Function Documentation

10.384.4.1 `template<typename G> void mln::util::vertex< G >::change_graph (const G & g) [inline]`

Change the parent graph of that vertex.

Definition at line 331 of file vertex.hh.

10.384.4.2 `template<typename G> edge< G > mln::util::vertex< G >::edge_with (const vertex< G > & v_id) const [inline]`

Returns true if this vertex has an edge with the given vertex.

Definition at line 321 of file vertex.hh.

10.384.4.3 `template<typename G> const G & mln::util::vertex< G >::graph () const [inline]`

Returns the graph pointer this vertex belongs to.

Definition at line 348 of file vertex.hh.

Referenced by mln::p_vertices< G, F >::has(), mln::util::line_graph< G >::has(), and mln::util::operator==().

10.384.4.4 `template<typename G> const vertex_id_t & mln::util::vertex< G >::id () const [inline]`

Returns the vertex id.

Definition at line 356 of file vertex.hh.

Referenced by mln::util::line_graph< G >::has(), and mln::util::operator==().

10.384.4.5 `template<typename G> void mln::util::vertex< G >::invalidate () [inline]`

Invalidate that vertex.

Definition at line 266 of file vertex.hh.

Referenced by mln::util::vertex< G >::vertex().

10.384.4.6 `template<typename G> bool mln::util::vertex< G >::is_valid () const [inline]`

Check whether the vertex is still part of the graph.

Definition at line 258 of file vertex.hh.

Referenced by mln::p_vertices< G, F >::has().

10.384.4.7 `template<typename G > edge_id_t mln::util::vertex< G >::ith_nbh_edge (unsigned i) const [inline]`

Returns the ith edge starting from this vertex.

Definition at line 285 of file vertex.hh.

10.384.4.8 `template<typename G > vertex_id_t mln::util::vertex< G >::ith_nbh_vertex (unsigned i) const [inline]`

Returns the ith vertex adjacent to this vertex.

Definition at line 303 of file vertex.hh.

10.384.4.9 `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_edges () const [inline]`

Returns the number max of edges starting from this vertex.

If g_ is a sub graph of another graph, nmax will be retrived from the initial graph.

Definition at line 294 of file vertex.hh.

10.384.4.10 `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_vertices () const [inline]`

Returns the number max of vertices adjacent to this vertex.

Definition at line 312 of file vertex.hh.

10.384.4.11 `template<typename G > mln::util::vertex< G >::operator vertex_id_t () const [inline]`

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

Definition at line 363 of file vertex.hh.

10.384.4.12 `template<typename G > vertex_id_t mln::util::vertex< G >::other (const edge_id_t & id_e) const [inline]`

Returns the other vertex located on edge id_e.

Definition at line 274 of file vertex.hh.

10.384.4.13 `template<typename G > void mln::util::vertex< G >::update_id (const vertex_id_t & id) [inline]`

Update the vertex id.

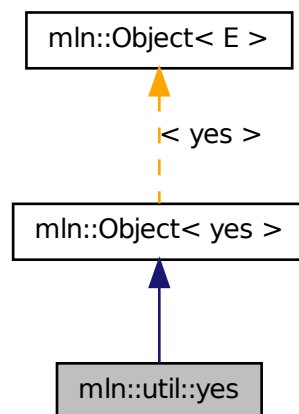
Definition at line 340 of file vertex.hh.

10.385 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for mln::util::yes:



10.385.1 Detailed Description

[Object](#) that always says "yes".

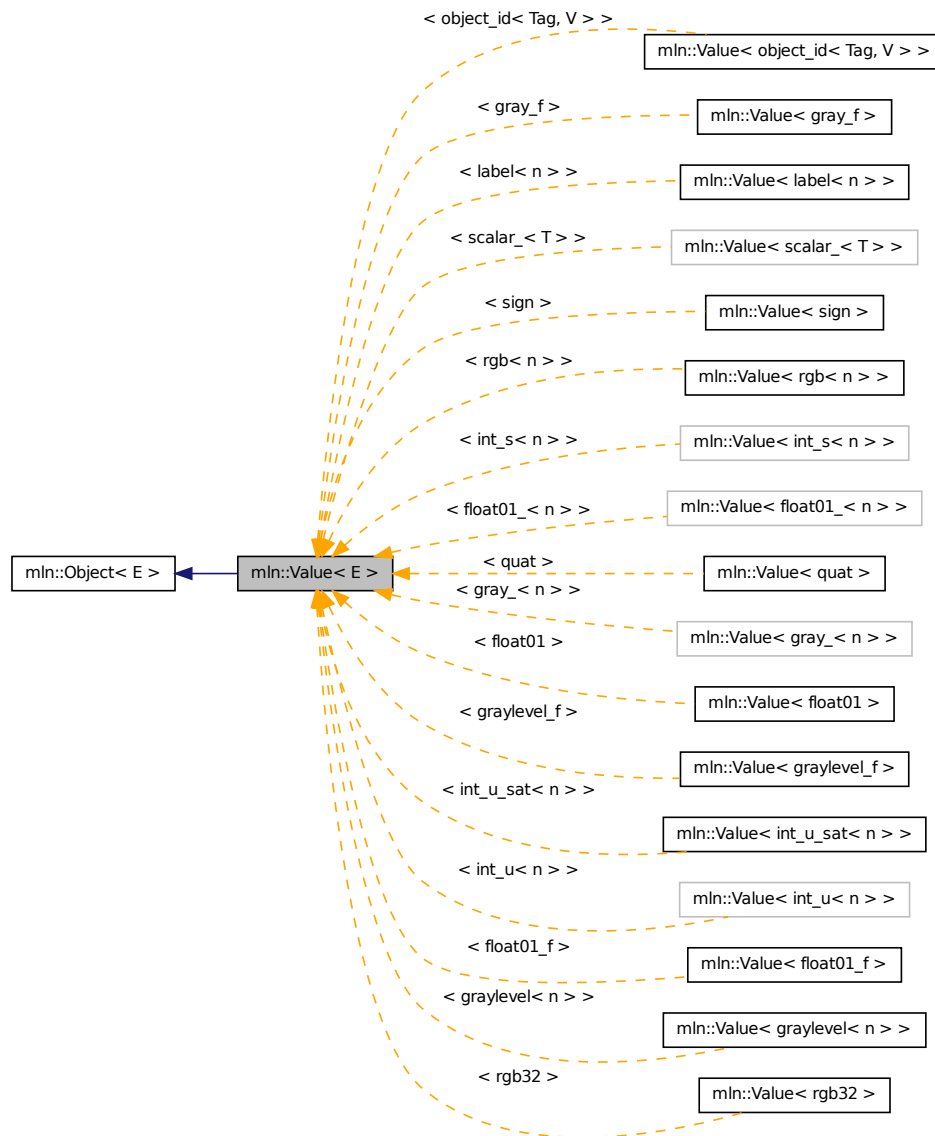
Definition at line 76 of file `yes.hh`.

10.386 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for `mln::Value< E >`:



10.386.1 Detailed Description

template<typename E> struct mln::Value< E >

Base class for implementation classes of values.

See also

`mln::doc::Value` for a complete documentation of this class contents.

Definition at line 57 of file core/concept/value.hh.

10.387 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0..1] and discretized with n bits.

```
#include <float01.hh>
```

Inherits mln::value::Floating< float01 >.

Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)
Encoding associated type.
- typedef float [equiv](#)
Equivalent associated type.

Public Member Functions

- [float01](#) ()
Ctor.
- template<unsigned n>
[float01](#) (const float01_< n > &val)
Ctor.
- [float01](#) (unsigned nbits, float val)
Ctor.
- unsigned [nbits](#) () const
Access to the encoding size.
- operator float () const
Conversion to float.
- [float01](#) & [set_nbits](#) (unsigned nbits)
Set the encoding size to nbits.
- const [float01](#) [to_nbits](#) (unsigned nbits) const
Return an equivalent gray encoded on nbits bits.
- float [value](#) () const
Access to std type.
- unsigned long [value_ind](#) () const
Access to the position in the quantized interval.

10.387.1 Detailed Description

Class for floating values restricted to the interval [0..1] and discretized with *n* bits.

Definition at line 56 of file float01.hh.

10.387.2 Member Typedef Documentation

10.387.2.1 `typedef std::pair<unsigned, unsigned long> mln::value::float01::enc`

Encoding associated type.

Definition at line 61 of file float01.hh.

10.387.2.2 `typedef float mln::value::float01::equiv`

Equivalent associated type.

Definition at line 64 of file float01.hh.

10.387.3 Constructor & Destructor Documentation

10.387.3.1 `mln::value::float01::float01 () [inline]`

Ctor.

Definition at line 151 of file float01.hh.

10.387.3.2 `template<unsigned n> mln::value::float01::float01 (const float01_<n> & val) [inline]`

Ctor.

Definition at line 158 of file float01.hh.

10.387.3.3 `mln::value::float01::float01 (unsigned nbits, float val) [inline]`

Ctor.

Definition at line 165 of file float01.hh.

10.387.4 Member Function Documentation

10.387.4.1 `unsigned mln::value::float01::nbits () const [inline]`

Access to the encoding size.

Definition at line 186 of file float01.hh.

10.387.4.2 `mln::value::float01::operator float () const [inline]`

Conversion to float.

Definition at line 224 of file float01.hh.

10.387.4.3 float01 & mln::value::float01::set_nbits (unsigned nbits) [inline]

Set the encoding size to nbits.

Definition at line 193 of file float01.hh.

Referenced by to_nbits().

10.387.4.4 const float01 mln::value::float01::to_nbits (unsigned nbits) const [inline]

Return an equivalent gray encoded on nbits bits.

Definition at line 214 of file float01.hh.

References set_nbits().

10.387.4.5 float mln::value::float01::value () const [inline]

Access to std type.

Definition at line 172 of file float01.hh.

10.387.4.6 unsigned long mln::value::float01::value_ind () const [inline]

Access to the position in the quantized interval.

Definition at line 179 of file float01.hh.

10.388 mln::value::float01_f Struct Reference

Class for floating values restricted to the interval [0..1].

```
#include <float01_f.hh>
```

Inherits mln::value::Floating< float01_f >, and mln::value::internal::value_like_< float,float,float,float01_f >.

Public Member Functions

- [float01_f](#) ()
Constructor without argument.
- [float01_f](#) (float val)
Constructor from a float.
- [operator float](#) () const
Conversion to a float.
- [float01_f](#) & [operator=](#) (const float val)
Assignment from a float.

- `float value () const`
Access to float value.

10.388.1 Detailed Description

Class for floating values restricted to the interval [0..1].

Definition at line 84 of file float01_f.hh.

10.388.2 Constructor & Destructor Documentation

10.388.2.1 `mln::value::float01_f::float01_f () [inline]`

Constructor without argument.

Definition at line 115 of file float01_f.hh.

10.388.2.2 `mln::value::float01_f::float01_f (float val) [inline]`

Constructor from a float.

Definition at line 120 of file float01_f.hh.

10.388.3 Member Function Documentation

10.388.3.1 `mln::value::float01_f::operator float () const [inline]`

Conversion to a float.

Definition at line 145 of file float01_f.hh.

10.388.3.2 `float01_f & mln::value::float01_f::operator= (const float val) [inline]`

Assignment from a float.

Definition at line 136 of file float01_f.hh.

10.388.3.3 `float mln::value::float01_f::value () const [inline]`

Access to float value.

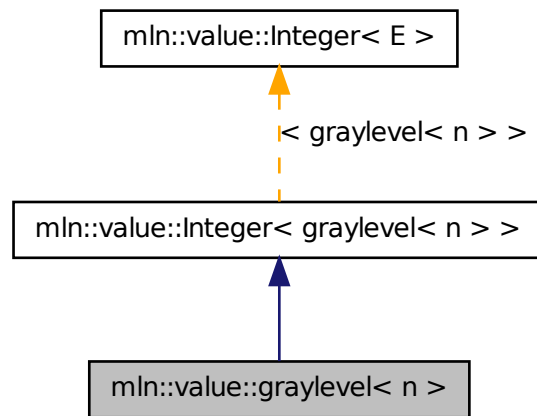
Definition at line 129 of file float01_f.hh.

10.389 `mln::value::graylevel< n >` Struct Template Reference

General gray-level class on n bits.

```
#include <graylevel.hh>
```

Inheritance diagram for mln::value::graylevel< n >:



Public Member Functions

- `graylevel ()`
Constructor without argument.
- `graylevel (const graylevel< n > &rhs)`
Copy constructor.
- `graylevel (int val)`
Constructor from int.
- `template<unsigned m>`
`graylevel (const graylevel< m > &rhs)`
Constructor from any graylevel.
- `graylevel< n > & operator= (const graylevel< n > &rhs)`
Assignment.
- `graylevel< n > & operator= (int val)`
Assignment with int.
- `template<unsigned m>`
`graylevel< n > & operator= (const graylevel< m > &rhs)`
Assignment with any graylevel.
- `float to_float () const`
Conversion to float between 0 and 1.

- unsigned [value](#) () const

Access to std type.

- [graylevel](#) (const [mln::literal::black_t](#) &)

Ctors with literals.

- [graylevel](#)< n > & [operator=](#) (const [mln::literal::black_t](#) &)

Assignment with literals.

10.389.1 Detailed Description

template<unsigned n> struct mln::value::graylevel< n >

General gray-level class on n bits.

Definition at line 257 of file graylevel.hh.

10.389.2 Constructor & Destructor Documentation

10.389.2.1 template<unsigned n> mln::value::graylevel< n >::graylevel () [inline]

Constructor without argument.

Definition at line 436 of file graylevel.hh.

10.389.2.2 template<unsigned n> mln::value::graylevel< n >::graylevel (const graylevel< n > & rhs) [inline]

Copy constructor.

Definition at line 463 of file graylevel.hh.

10.389.2.3 template<unsigned n> mln::value::graylevel< n >::graylevel (int val) [inline]

Constructor from int.

Definition at line 443 of file graylevel.hh.

10.389.2.4 template<unsigned n> template<unsigned m> mln::value::graylevel< n >::graylevel (const graylevel< m > & rhs) [inline]

Constructor from any graylevel.

Definition at line 481 of file graylevel.hh.

References [mln::value::graylevel< n >::value\(\)](#).

10.389.2.5 `template<unsigned n> mln::value::graylevel< n >::graylevel (const
mln::literal::black_t &) [inline]`

Ctors with literals.

Definition at line 499 of file graylevel.hh.

10.389.3 Member Function Documentation

10.389.3.1 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const
graylevel< n > & rhs) [inline]`

Assignment.

Definition at line 472 of file graylevel.hh.

10.389.3.2 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (int
val) [inline]`

Assignment with int.

Definition at line 453 of file graylevel.hh.

10.389.3.3 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const
mln::literal::black_t &) [inline]`

Assignment with literals.

Definition at line 507 of file graylevel.hh.

10.389.3.4 `template<unsigned n> template<unsigned m> graylevel< n > &
mln::value::graylevel< n >::operator= (const graylevel< m > & rhs) [inline]`

Assignment with any graylevel.

Definition at line 490 of file graylevel.hh.

References mln::value::graylevel< n >::value().

10.389.3.5 `template<unsigned n> float mln::value::graylevel< n >::to_float () const
[inline]`

Conversion to float between 0 and 1.

Definition at line 557 of file graylevel.hh.

Referenced by mln::value::graylevel_f::graylevel_f(), and mln::value::graylevel_f::operator=().

10.389.3.6 `template<unsigned n> unsigned mln::value::graylevel< n >::value () const
[inline]`

Access to std type.

Definition at line 549 of file graylevel.hh.

Referenced by `mln::value::graylevel< n >::graylevel()`, and `mln::value::graylevel< n >::operator=()`.

10.390 mln::value::graylevel_f Struct Reference

General gray-level class on n bits.

```
#include <graylevel_f.hh>
```

Inherits `mln::value::Floating< graylevel_f >`, and `mln::value::internal::value_like_< float01_f, float01_f::enc, internal::gray_f, graylevel_f >`.

Public Member Functions

- [graylevel_f \(\)](#)
Constructor without argument.
- [graylevel_f \(const graylevel_f &rhs\)](#)
Copy constructor.
- [graylevel_f \(float val\)](#)
Constructor from float.
- `template<unsigned n>`
[graylevel_f \(const graylevel< n > &rhs\)](#)
Constructor from graylevel.
- `template<unsigned n>`
[operator graylevel< n > \(\) const](#)
Conversion to graylevel<n>.
- [graylevel_f & operator= \(float val\)](#)
Assignment with float.
- [graylevel_f & operator= \(const graylevel_f &rhs\)](#)
Assignment.
- `template<unsigned n>`
[graylevel_f & operator= \(const graylevel< n > &rhs\)](#)
Assignment with graylevel.
- `float value () const`
Access to std type.
- [graylevel_f \(const mln::literal::black_t &\)](#)
Ctors with literals.
- [graylevel_f & operator= \(const mln::literal::black_t &\)](#)
Assignment with literals.

10.390.1 Detailed Description

General gray-level class on n bits.

Definition at line 193 of file graylevel_f.hh.

10.390.2 Constructor & Destructor Documentation

10.390.2.1 mln::value::graylevel_f::graylevel_f () [inline]

Constructor without argument.

Definition at line 341 of file graylevel_f.hh.

10.390.2.2 mln::value::graylevel_f::graylevel_f (const graylevel_f & rhs) [inline]

Copy constructor.

Definition at line 383 of file graylevel_f.hh.

10.390.2.3 mln::value::graylevel_f::graylevel_f (float val) [inline]

Constructor from float.

Definition at line 347 of file graylevel_f.hh.

10.390.2.4 template<unsigned n> mln::value::graylevel_f::graylevel_f (const graylevel< n > & rhs)

Constructor from graylevel.

Definition at line 365 of file graylevel_f.hh.

References mln::value::graylevel< n >::to_float().

10.390.2.5 mln::value::graylevel_f::graylevel_f (const mln::literal::black_t &) [inline]

Ctors with literals.

Definition at line 400 of file graylevel_f.hh.

10.390.3 Member Function Documentation

10.390.3.1 template<unsigned n> mln::value::graylevel_f::operator graylevel< n > () const [inline]

Conversion to graylevel<n>.

Definition at line 443 of file graylevel_f.hh.

10.390.3.2 graylevel_f & mln::value::graylevel_f::operator= (float val) [inline]

Assignment with float.

Definition at line 356 of file graylevel_f.hh.

10.390.3.3 `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= (const graylevel< n > & rhs)`

Assignment with graylevel.

Definition at line 374 of file graylevel_f.hh.

References `mln::value::graylevel< n >::to_float()`.

10.390.3.4 `graylevel_f & mln::value::graylevel_f::operator= (const mln::literal::black_t &)`
`[inline]`

Assignment with literals.

Definition at line 407 of file graylevel_f.hh.

10.390.3.5 `graylevel_f & mln::value::graylevel_f::operator= (const graylevel_f & rhs)`
`[inline]`

Assignment.

Definition at line 391 of file graylevel_f.hh.

10.390.3.6 `float mln::value::graylevel_f::value () const` `[inline]`

Access to std type.

Definition at line 450 of file graylevel_f.hh.

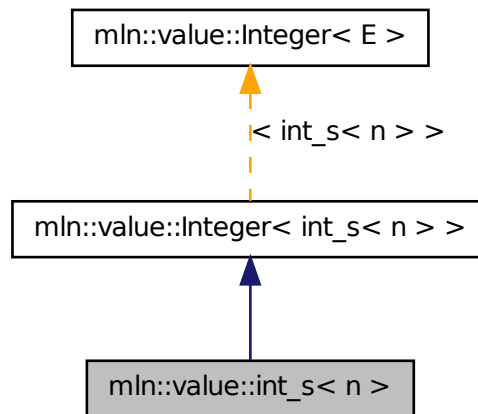
Referenced by `mln::value::operator<<()`.

10.391 `mln::value::int_s< n >` Struct Template Reference

Signed integer value class.

```
#include <int_s.hh>
```

Inheritance diagram for mln::value::int_s< n >:



Public Member Functions

- `int_s ()`
Constructor without argument.
- `int_s (int i)`
Constructor from an integer.
- `operator int () const`
Conversion to an integer.
- `int_s< n > & operator= (int i)`
Assignment from an integer.
- `int_s (const mln::literal::zero_t &)`
Constructors/assignments with literals.

Static Public Attributes

- static const `int_s< n > one = 1`
Unit value.
- static const `int_s< n > zero = 0`
Zero value.

10.391.1 Detailed Description

template<unsigned n> struct mln::value::int_s< n >

Signed integer value class. The parameter is *n* the number of encoding bits.

Definition at line 115 of file int_s.hh.

10.391.2 Constructor & Destructor Documentation

10.391.2.1 template<unsigned n> mln::value::int_s< n >::int_s() [inline]

Constructor without argument.

Definition at line 179 of file int_s.hh.

10.391.2.2 template<unsigned n> mln::value::int_s< n >::int_s(int i) [inline]

Constructor from an integer.

Definition at line 192 of file int_s.hh.

**10.391.2.3 template<unsigned n> mln::value::int_s< n >::int_s(const mln::literal::zero_t &)
 [inline]**

Constructors/assignments with literals.

Definition at line 222 of file int_s.hh.

10.391.3 Member Function Documentation

10.391.3.1 template<unsigned n> mln::value::int_s< n >::operator int() const [inline]

Conversion to an integer.

Definition at line 185 of file int_s.hh.

**10.391.3.2 template<unsigned n> int_s< n > & mln::value::int_s< n >::operator=(int i)
 [inline]**

Assignment from an integer.

Definition at line 207 of file int_s.hh.

10.391.4 Member Data Documentation

10.391.4.1 template<unsigned n> const int_s< n > mln::value::int_s< n >::one = 1 [static]

Unit value.

Definition at line 149 of file int_s.hh.

10.391.4.2 `template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0 [static]`

Zero value.

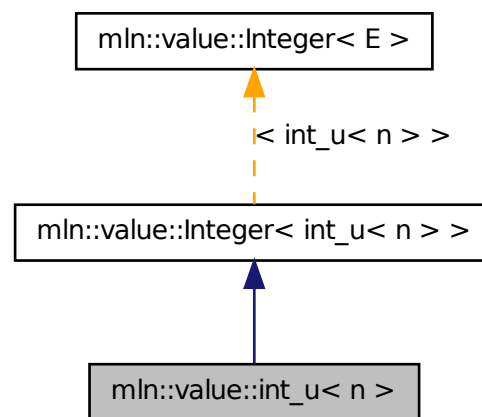
Definition at line 146 of file int_s.hh.

10.392 mln::value::int_u< n > Struct Template Reference

Unsigned integer value class.

```
#include <int_u.hh>
```

Inheritance diagram for mln::value::int_u< n >:



Public Member Functions

- `int_u ()`
Constructor without argument.
- `int_u (int i)`
Constructor from an integer.
- `int_u< n > next () const`
Give the next value (i.e., $i + 1$).
- `operator unsigned () const`
Conversion to an unsigned integer.
- `int operator- () const`
Unary operator minus.

- `int_u< n > & operator= (int i)`
Assignment from an integer.
- `int_u (const mln::literal::zero_t &)`
Constructors/assignments with literals.

10.392.1 Detailed Description

template<unsigned n> struct mln::value::int_u< n >

Unsigned integer value class. The parameter is `n` the number of encoding bits.

Definition at line 156 of file `int_u.hh`.

10.392.2 Constructor & Destructor Documentation

10.392.2.1 template<unsigned n> mln::value::int_u< n >::int_u () [inline]

Constructor without argument.

Definition at line 276 of file `int_u.hh`.

10.392.2.2 template<unsigned n> mln::value::int_u< n >::int_u (int i) [inline]

Constructor from an integer.

Definition at line 282 of file `int_u.hh`.

10.392.2.3 template<unsigned n> mln::value::int_u< n >::int_u (const mln::literal::zero_t &) [inline]

Constructors/assignments with literals.

Definition at line 291 of file `int_u.hh`.

10.392.3 Member Function Documentation

10.392.3.1 template<unsigned n> int_u< n > mln::value::int_u< n >::next () const [inline]

Give the next value (i.e., `i + 1`).

Definition at line 350 of file `int_u.hh`.

10.392.3.2 template<unsigned n> mln::value::int_u< n >::operator unsigned () const [inline]

Conversion to an unsigned integer.

Definition at line 323 of file int_u.hh.

10.392.3.3 `template<unsigned n> int mln::value::int_u< n >::operator-() const [inline]`

Unary operator minus.

Definition at line 331 of file int_u.hh.

10.392.3.4 `template<unsigned n> int_u< n > & mln::value::int_u< n >::operator=(int i) [inline]`

Assignment from an integer.

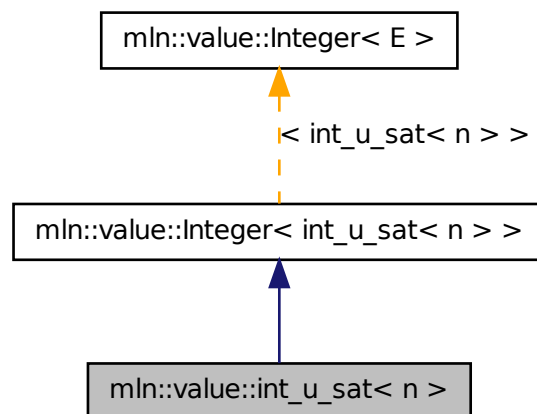
Definition at line 339 of file int_u.hh.

10.393 mln::value::int_u_sat< n > Struct Template Reference

Unsigned integer value class with saturation behavior.

`#include <int_u_sat.hh>`

Inheritance diagram for mln::value::int_u_sat< n >:



Public Member Functions

- `int_u_sat()`
Constructor without argument.
- `int_u_sat(int i)`
Constructor from an integer.

- `operator int () const`
Conversion to an integer.
- `int_u_sat< n > & operator+= (int i)`
Self addition.
- `int_u_sat< n > & operator-= (int i)`
Self subtraction.
- `int_u_sat< n > & operator= (int i)`
Assignment from an integer.

Static Public Attributes

- static const `int_u_sat< n > one = 1`
Unit value.
- static const `int_u_sat< n > zero = 0`
Zero value.

10.393.1 Detailed Description

`template<unsigned n> struct mln::value::int_u_sat< n >`

Unsigned integer value class with saturation behavior. The parameter is `n` the number of encoding bits.
Definition at line 90 of file `int_u_sat.hh`.

10.393.2 Constructor & Destructor Documentation

10.393.2.1 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat () [inline]`

Constructor without argument.

Definition at line 149 of file `int_u_sat.hh`.

10.393.2.2 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat (int i) [inline]`

Constructor from an integer.

Definition at line 155 of file `int_u_sat.hh`.

10.393.3 Member Function Documentation

10.393.3.1 `template<unsigned n> mln::value::int_u_sat< n >::operator int () const [inline]`

Conversion to an integer.

Definition at line 170 of file int_u_sat.hh.

10.393.3.2 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+=(
int i) [inline]`

Self addition.

Definition at line 195 of file int_u_sat.hh.

10.393.3.3 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-=(
int i) [inline]`

Self subtraction.

Definition at line 205 of file int_u_sat.hh.

10.393.3.4 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator=(int
i) [inline]`

Assignment from an integer.

Definition at line 178 of file int_u_sat.hh.

10.393.4 Member Data Documentation

10.393.4.1 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1
[static]`

Unit value.

Definition at line 115 of file int_u_sat.hh.

10.393.4.2 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0
[static]`

Zero value.

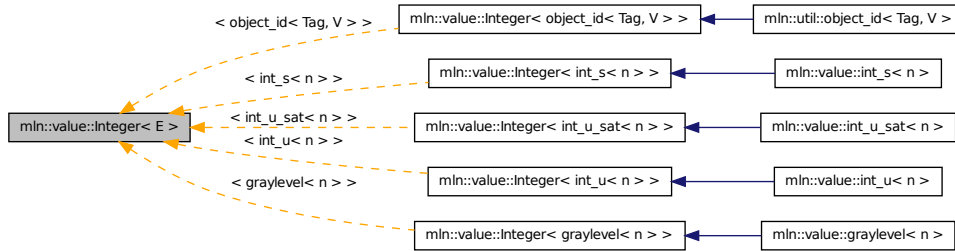
Definition at line 112 of file int_u_sat.hh.

10.394 mln::value::Integer< E > Struct Template Reference

Concept of integer.

`#include <integer.hh>`

Inheritance diagram for `mln::value::Integer< E >`:



10.394.1 Detailed Description

template<typename E> struct mln::value::Integer< E >

Concept of integer.

Definition at line 58 of file `concept/integer.hh`.

10.395 mln::value::Integer< void > Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

10.395.1 Detailed Description

template<> struct mln::value::Integer< void >

Category flag type.

Definition at line 50 of file `concept/integer.hh`.

10.396 mln::value::label< n > Struct Template Reference

Label value class.

```
#include <label.hh>
```

Inherits `mln::value::Symbolic< label< n > >`, and `mln::value::internal::value_like< unsigned,internal::encoding_unsigned< n >::ret,int,label< n > >`.

Public Types

- `typedef internal::encoding_unsigned< n >::ret enc`
Encoding associated type.

Public Member Functions

- [label](#) ()
Constructor without argument.
- [label](#) (unsigned i)
Constructor from an (unsigned) integer.
- [label](#) (const [literal::zero_t](#) &v)
Constructor from [literal::zero](#).
- [label](#)< n > [next](#) () const
Return the next value.
- [operator unsigned](#) () const
Conversion to an unsigned integer.
- [label](#)< n > & [operator++](#) ()
Self increment.
- [label](#)< n > & [operator--](#) ()
Self decrement.
- [label](#)< n > & [operator=](#) (unsigned i)
Assignment from an (unsigned) integer.
- [label](#)< n > & [operator=](#) (const [literal::zero_t](#) &v)
Assignment from [literal::zero](#).
- [label](#)< n > [prev](#) () const
Return the previous value.

10.396.1 Detailed Description

template<unsigned n> struct mln::value::label< n >

Label value class. The parameter n is the number of encoding bits.

Definition at line 140 of file label.hh.

10.396.2 Member Typedef Documentation

**10.396.2.1 template<unsigned n> typedef internal::encoding_unsigned_<n>::ret
 mln::value::label< n >::enc**

Encoding associated type.

Definition at line 150 of file label.hh.

10.396.3 Constructor & Destructor Documentation

10.396.3.1 `template<unsigned n> mln::value::label< n >::label () [inline]`

Constructor without argument.

Definition at line 271 of file label.hh.

10.396.3.2 `template<unsigned n> mln::value::label< n >::label (unsigned i) [inline]`

Constructor from an (unsigned) integer.

Definition at line 277 of file label.hh.

10.396.3.3 `template<unsigned n> mln::value::label< n >::label (const literal::zero_t & v) [inline]`

Constructor from [literal::zero](#).

Definition at line 284 of file label.hh.

10.396.4 Member Function Documentation

10.396.4.1 `template<unsigned n> label< n > mln::value::label< n >::next () const [inline]`

Return the next value.

Definition at line 338 of file label.hh.

10.396.4.2 `template<unsigned n> mln::value::label< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

Definition at line 291 of file label.hh.

10.396.4.3 `template<unsigned n> label< n > & mln::value::label< n >::operator++ () [inline]`

Self increment.

Definition at line 318 of file label.hh.

10.396.4.4 `template<unsigned n> label< n > & mln::value::label< n >::operator-- () [inline]`

Self decrement.

Definition at line 328 of file label.hh.

10.396.4.5 `template<unsigned n> label< n > & mln::value::label< n >::operator= (unsigned i) [inline]`

Assignment from an (unsigned) integer.

Definition at line 299 of file label.hh.

10.396.4.6 `template<unsigned n> label< n > & mln::value::label< n >::operator= (const literal::zero_t & v) [inline]`

Assignment from [literal::zero](#).

Definition at line 309 of file label.hh.

10.396.4.7 `template<unsigned n> label< n > mln::value::label< n >::prev () const [inline]`

Return the previous value.

Definition at line 346 of file label.hh.

10.397 mln::value::lut_vec< S, T > Struct Template Reference

Class that defines FIXME.

```
#include <lut_vec.hh>
```

Inherits Value_Set< lut_vec< S, T > >.

Public Types

- `typedef bkd_viter_< lut_vec< S, T > > bkd_viter`
Backward Value_Iterator associated type.
- `typedef fwd_viter_< lut_vec< S, T > > fwd_viter`
Forward Value_Iterator associated type.
- `typedef T value`
Value associated type.

Public Member Functions

- `bool has (const value &v) const`
Test if v belongs to this set.
- `unsigned index_of (const value &v) const`
Give the index of value v in this set.
- `unsigned nvalues () const`

Give the number of values.

- `T operator[] (unsigned i) const`
Give the i -th value.
- `template<typename F >`
`lut_vec (const S &vset, const Function_v2v< F > &f)`
Constructors
Constructor from a value set and any Function_v2v.
- `template<typename V >`
`lut_vec (const S &vset, const Function_v2v< fun::i2v::array< V > > &f)`
Constructor from a value set and any fun::i2v::array.
- `template<typename V >`
`lut_vec (const S &vset, const Function_v2v< util::array< V > > &f)`
Constructor from a value set and any util::array.

10.397.1 Detailed Description

`template<typename S, typename T> struct mln::value::lut_vec< S, T >`

Class that defines FIXME.

Warning

This is a multi-set!!! FIXME

Definition at line 71 of file lut_vec.hh.

10.397.2 Member Typedef Documentation

10.397.2.1 `template<typename S , typename T > typedef bkd_viter_< lut_vec<S,T> >`
`mln::value::lut_vec< S, T >::bkd_viter`

Backward Value_Iterator associated type.

Definition at line 80 of file lut_vec.hh.

10.397.2.2 `template<typename S , typename T > typedef fwd_viter_< lut_vec<S,T> >`
`mln::value::lut_vec< S, T >::fwd_viter`

Forward Value_Iterator associated type.

Definition at line 77 of file lut_vec.hh.

10.397.2.3 `template<typename S , typename T > typedef T mln::value::lut_vec< S, T >::value`

`Value` associated type.

Definition at line 74 of file lut_vec.hh.

10.397.3 Constructor & Destructor Documentation

10.397.3.1 `template<typename S , typename T > template<typename F > mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< F > & f) [inline]`

Constructors

Constructor from a value set and any [Function_v2v](#).

Definition at line 148 of file lut_vec.hh.

10.397.3.2 `template<typename S , typename T > template<typename V > mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< fun::i2v::array< V > > & f) [inline]`

Constructor from a value set and any [fun::i2v::array](#).

Definition at line 161 of file lut_vec.hh.

10.397.3.3 `template<typename S , typename T > template<typename V > mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< util::array< V > > & f) [inline]`

Constructor from a value set and any [util::array](#).

Definition at line 172 of file lut_vec.hh.

References [mln::util::array< T >::size\(\)](#), and [mln::util::array< T >::std_vector\(\)](#).

10.397.4 Member Function Documentation

10.397.4.1 `template<typename S , typename T > bool mln::value::lut_vec< S, T >::has (const value & v) const`

Test if v belongs to this set.

10.397.4.2 `template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::index_of (const value & v) const`

Give the index of value v in this set.

10.397.4.3 `template<typename S , typename T > unsigned mln::value::lut_vec< S, T >::nvalues () const [inline]`

Give the number of values.

Definition at line 202 of file lut_vec.hh.

Referenced by [mln::value::lut_vec< S, T >::operator\[\]\(\)](#).

10.397.4.4 `template<typename S , typename T > T mln::value::lut_vec< S, T >::operator[] (unsigned i) const [inline]`

Give the `i-th` value.

Definition at line 193 of file `lut_vec.hh`.

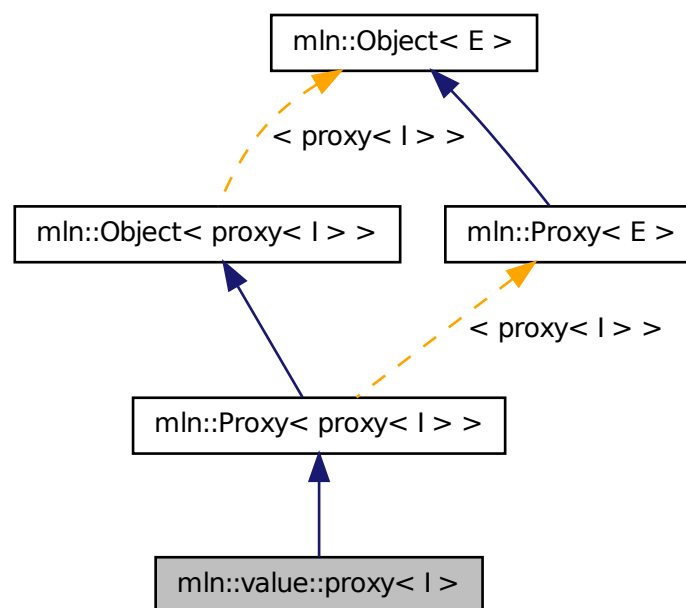
References `mln::value::lut_vec< S, T >::nvalues()`.

10.398 `mln::value::proxy< I >` Class Template Reference

Generic proxy class for an image pixel value.

```
#include <proxy.hh>
```

Inheritance diagram for `mln::value::proxy< I >`:



Public Types

- typedef void [enc](#)
Encoding associated type.
- typedef I::value [equiv](#)
Equivalent associated type.

Public Member Functions

- `proxy< I > & operator=` (const `proxy< I >` &rhs)
Assignment (write access); replacement for default op.
- `template<typename J > proxy< I > & operator=` (const `proxy< J >` &rhs)
Assignment (write access); with other proxy.
- `proxy ()`
Constructor.
- `proxy (I &ima, const typename I::psite &p)`
Constructor.
- `I::value to_value ()` const
Explicit read access.
- `~proxy ()`
Destructor.

10.398.1 Detailed Description

`template<typename I> class mln::value::proxy< I >`

Generic proxy class for an image pixel value. The parameter `I` is an image type.

Definition at line 85 of file `value/proxy.hh`.

10.398.2 Member Typedef Documentation

10.398.2.1 `template<typename I> typedef void mln::value::proxy< I >::enc`

Encoding associated type.

Definition at line 91 of file `value/proxy.hh`.

10.398.2.2 `template<typename I> typedef I::value mln::value::proxy< I >::equiv`

Equivalent associated type.

Definition at line 94 of file `value/proxy.hh`.

10.398.3 Constructor & Destructor Documentation

10.398.3.1 `template<typename I> mln::value::proxy< I >::proxy () [inline]`

Constructor.

Definition at line 150 of file `value/proxy.hh`.

10.398.3.2 `template<typename I> mln::value::proxy<I>::proxy (I & ima, const typename I::psite & p) [inline]`

Constructor.

Definition at line 157 of file value/proxy.hh.

10.398.3.3 `template<typename I> mln::value::proxy<I>::~~proxy () [inline]`

Destructor.

Definition at line 165 of file value/proxy.hh.

10.398.4 Member Function Documentation

10.398.4.1 `template<typename I> proxy<I> & mln::value::proxy<I>::operator= (const proxy<I> & rhs) [inline]`

Assignment (write access); replacement for default op.

Definition at line 186 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.398.4.2 `template<typename I> template<typename J> proxy<I> & mln::value::proxy<I>::operator= (const proxy<J> & rhs) [inline]`

Assignment (write access); with other proxy.

Definition at line 199 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.398.4.3 `template<typename I> I::value mln::value::proxy<I>::to_value () const [inline]`

Explicit read access.

Definition at line 226 of file value/proxy.hh.

Referenced by `mln::value::proxy<I>::operator=()`.

10.399 mln::value::qt::rgb32 Struct Reference

Color class for red-green-blue where every component is n-bit encoded.

`#include <rgb32.hh>`

Inherits `mln::value::Vectorial<rgb32>`, and `mln::value::internal::value_like_<algebra::vec<3, int_u<8>>, algebra::vec<3, int_u<8>>, algebra::vec<3, int>>, rgb32>`.

Public Member Functions

- `rgb32 & operator= (const rgb32 &rhs)`

Assignment.

- `rgb32` (`const algebra::vec< 3, int > &rhs`)

Constructor from a algebra::vec.

- `rgb32` (`int r, int g, int b`)

Constructor from component values.

- `rgb32` ()

Constructor without argument.

- `int_u< 8 > red` () `const`

Acces to red/green/blue component.

- `rgb32` (`const mln::literal::zero_t &`)

Constructors with literals.

Static Public Attributes

- `static const rgb32 zero`

Zero value.

10.399.1 Detailed Description

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 197 of file rgb32.hh.

10.399.2 Constructor & Destructor Documentation

10.399.2.1 `mln::value::qt::rgb32::rgb32 () [inline]`

Constructor without argument.

Definition at line 385 of file rgb32.hh.

10.399.2.2 `mln::value::qt::rgb32::rgb32 (int r, int g, int b) [inline]`

Constructor from component values.

Definition at line 427 of file rgb32.hh.

10.399.2.3 `mln::value::qt::rgb32::rgb32 (const algebra::vec< 3, int > & rhs) [inline]`

Constructor from a algebra::vec.

Definition at line 391 of file rgb32.hh.

10.399.2.4 `mln::value::qt::rgb32::rgb32 (const mln::literal::zero_t &) [inline]`

Constructors with literals.

Definition at line 442 of file `rgb32.hh`.

10.399.3 Member Function Documentation

10.399.3.1 `rgb32 & mln::value::qt::rgb32::operator= (const rgb32 & rhs) [inline]`

Assignment.

Definition at line 623 of file `rgb32.hh`.

10.399.3.2 `int_u<8> mln::value::qt::rgb32::red () const [inline]`

Acces to red/green/blue component.

Definition at line 212 of file `rgb32.hh`.

10.399.4 Member Data Documentation

10.399.4.1 `const rgb32 mln::value::qt::rgb32::zero [static]`

Zero value.

Definition at line 272 of file `rgb32.hh`.

10.400 `mln::value::rgb< n >` Struct Template Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb.hh>
```

Inherits `mln::value::Vectorial< rgb< n > >`, and `mln::value::internal::value_like_< algebra::vec< 3, int_u< n > >, algebra::vec< 3, int_u< n > >, algebra::vec< 3, int >, rgb< n > >`.

Public Member Functions

- `rgb< n > & operator= (const rgb< n > &rhs)`

Assignment.

- `rgb (const algebra::vec< 3, int > &rhs)`

Constructor from a algebra::vec.

- `rgb (int r, int g, int b)`

Constructor from component values.

- `rgb ()`

Constructor without argument.

- `int_u< n > red () const`
Acces to red/green/blue component.
- `rgb (const mln::literal::white_t &)`
Constructors with literals.

Static Public Attributes

- static const `rgb< n > zero`
Zero value.

10.400.1 Detailed Description

`template<unsigned n> struct mln::value::rgb< n >`

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 248 of file value/rgb.hh.

10.400.2 Constructor & Destructor Documentation

10.400.2.1 `template<unsigned n> mln::value::rgb< n >::rgb () [inline]`

Constructor without argument.

Definition at line 422 of file value/rgb.hh.

10.400.2.2 `template<unsigned n> mln::value::rgb< n >::rgb (int r, int g, int b) [inline]`

Constructor from component values.

Definition at line 458 of file value/rgb.hh.

10.400.2.3 `template<unsigned n> mln::value::rgb< n >::rgb (const algebra::vec< 3, int > & rhs) [inline]`

Constructor from a algebra::vec.

Definition at line 428 of file value/rgb.hh.

10.400.2.4 `template<unsigned n> mln::value::rgb< n >::rgb (const mln::literal::white_t &) [inline]`

Constructors with literals.

Definition at line 473 of file value/rgb.hh.

10.400.3 Member Function Documentation

10.400.3.1 `template<unsigned n> rgb< n > & mln::value::rgb< n >::operator= (const rgb< n > & rhs) [inline]`

Assignment.

Definition at line 645 of file value/rgb.hh.

10.400.3.2 `template<unsigned n> int_u<n> mln::value::rgb< n >::red () const [inline]`

Acces to red/green/blue component.

Definition at line 264 of file value/rgb.hh.

Referenced by `mln::fun::v2v::rgb8_to_rgbn< n >::operator()`.

10.400.4 Member Data Documentation

10.400.4.1 `template<unsigned n> const rgb< n > mln::value::rgb< n >::zero [static]`

Zero value.

Definition at line 322 of file value/rgb.hh.

10.401 mln::value::set< T > Struct Template Reference

Class that defines the set of values of type T.

```
#include <set.hh>
```

Inherits `set_selector_< T, set< T >, mln::metal::equal< mln::trait::value_< T >::quant, mln::trait::value::quant::low >::value >`.

Static Public Member Functions

- static const `set< T > & the ()`

Return a singleton.

10.401.1 Detailed Description

`template<typename T> struct mln::value::set< T >`

Class that defines the set of values of type T. This is the exhaustive set of values obtainable from type T.

Definition at line 65 of file value/set.hh.

10.401.2 Member Function Documentation

10.401.2.1 `template<typename T> const set< T> & mln::value::set< T>::the () [inline, static]`

Return a singleton.

Definition at line 80 of file value/set.hh.

10.402 mln::value::sign Class Reference

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

```
#include <sign.hh>
```

Inherits Integer< sign >.

Public Types

- typedef int `enc`
FIXME Are these typedefs correct?
- typedef int `equiv`
Define the equivalent type.

Public Member Functions

- `operator int () const`
Conversion to an integer.
- `sign & operator= (int i)`
Assignment from an integer.
- `sign ()`
Constructor without argument.
- `sign (int i)`
Constructor from an integer.
- `sign (const mln::literal::zero_t &)`
Constructors/assignments with literals.

Static Public Attributes

- static const `sign one = 1`
Unit value.

- static const `sign zero` = 0

Zero value.

10.402.1 Detailed Description

The `sign` class represents the value type composed by the set (-1, 0, 1) `sign` value type is a subset of the `int` value type.

Definition at line 49 of file `value/sign.hh`.

10.402.2 Member Typedef Documentation

10.402.2.1 `typedef int mln::value::sign::enc`

FIXME Are these typedefs correct?

Define the encoding type

Definition at line 55 of file `value/sign.hh`.

10.402.2.2 `typedef int mln::value::sign::equiv`

Define the equivalent type.

Definition at line 58 of file `value/sign.hh`.

10.402.3 Constructor & Destructor Documentation

10.402.3.1 `mln::value::sign::sign () [inline]`

Constructor without argument.

Definition at line 119 of file `value/sign.hh`.

10.402.3.2 `mln::value::sign::sign (int i) [inline]`

Constructor from an integer.

Definition at line 137 of file `value/sign.hh`.

10.402.3.3 `mln::value::sign::sign (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 155 of file `value/sign.hh`.

10.402.4 Member Function Documentation

10.402.4.1 mln::value::sign::operator int () const [inline]

Conversion to an integer.

Definition at line 124 of file value/sign.hh.

10.402.4.2 sign & mln::value::sign::operator= (int i) [inline]

Assignment from an integer.

Definition at line 146 of file value/sign.hh.

10.402.5 Member Data Documentation

10.402.5.1 const sign mln::value::sign::one = 1 [static]

Unit value.

Definition at line 88 of file value/sign.hh.

10.402.5.2 const sign mln::value::sign::zero = 0 [static]

Zero value.

Definition at line 85 of file value/sign.hh.

10.403 mln::value::stack_image< n, I > Struct Template Reference

Stack image class.

```
#include <stack.hh>
```

Inherits image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >.

Public Types

- typedef I::domain_t [domain_t](#)
Site_Set associated type.
- typedef internal::helper_stack_image_lvalue_< n, I >::ret [lvalue](#)
Return type of read-write access.
- typedef I::psite [psite](#)
Point_Site associated type.
- typedef [value](#) [rvalue](#)
Return type of read-only access.
- typedef [stack_image](#)< n, tag::image_< I > > [skeleton](#)

Skeleton.

- `typedef algebra::vec< n, typename I::value > value`
Value associated type.

Public Member Functions

- `bool is_valid () const`
Test if this image has been initialized.
- `lvalue operator() (const psite &)`
Read-write access of pixel value at point site p.
- `rvalue operator() (const psite &p) const`
Read-only access of pixel value at point site p.
- `stack_image (const algebra::vec< n, I > &imas)`
Constructors.

10.403.1 Detailed Description

`template<unsigned n, typename I> struct mln::value::stack_image< n, I >`

Stack image class. `mln::value::stack_image` stores a vector of n images of the same domain.

The parameter n is the number of images, I is the type of a stack element. Acces a value will compute a vector which contains n coordinates : [stack[0](p), stack[1](p), ... , stack[n](p)]

Definition at line 145 of file stack.hh.

10.403.2 Member Typedef Documentation

10.403.2.1 `template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t`

`Site_Set` associated type.

Definition at line 154 of file stack.hh.

10.403.2.2 `template<unsigned n, typename I> typedef internal::helper_stack_image_lvalue<n,I>::ret mln::value::stack_image< n, I >::lvalue`

Return type of read-write access.

Definition at line 166 of file stack.hh.

10.403.2.3 `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

Point_Site associated type.

Definition at line 151 of file stack.hh.

10.403.2.4 `template<unsigned n, typename I> typedef value mln::value::stack_image< n, I >::rvalue`

Return type of read-only access.

The rvalue type is not a const reference, since the value type is built on the fly, and return by value (copy).

Definition at line 163 of file stack.hh.

10.403.2.5 `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton`

Skeleton.

Definition at line 170 of file stack.hh.

10.403.2.6 `template<unsigned n, typename I> typedef algebra::vec<n, typename I ::value> mln::value::stack_image< n, I >::value`

[Value](#) associated type.

Definition at line 157 of file stack.hh.

10.403.3 Constructor & Destructor Documentation

10.403.3.1 `template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image (const algebra::vec< n, I > & imas) [inline]`

Constructors.

Definition at line 236 of file stack.hh.

10.403.4 Member Function Documentation

10.403.4.1 `template<unsigned n, typename I> bool mln::value::stack_image< n, I >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 255 of file stack.hh.

10.403.4.2 `template<unsigned n, typename I> stack_image< n, I >::lvalue mln::value::stack_image< n, I >::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site p.

Definition at line 296 of file stack.hh.

10.403.4.3 `template<unsigned n, typename I> stack_image< n, I>::rvalue
mln::value::stack_image< n, I>::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 277 of file stack.hh.

10.404 mln::value::super_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

10.404.1 Detailed Description

`template<> struct mln::value::super_value< sign >`

Specializations: Sign type is a subset of the short value type.

Definition at line 56 of file super_value.hh.

10.405 mln::value::value_array< T, V > Struct Template Reference

Generic array class over indexed by a value set with type T.

```
#include <value_array.hh>
```

Public Member Functions

- `const V & operator() (const T &v) const`
}
- `const V & operator[] (unsigned i) const`
}
- `value_array ()`
Constructors.
- `const mln::value::set< T > & vset () const`
}

10.405.1 Detailed Description

`template<typename T, typename V> struct mln::value::value_array< T, V >`

Generic array class over indexed by a value set with type T.

Definition at line 45 of file value_array.hh.

10.405.2 Constructor & Destructor Documentation

10.405.2.1 `template<typename T , typename V > mln::value::value_array< T, V >::value_array
() [inline]`

Constructors.

```
{
```

Definition at line 89 of file value_array.hh.

10.405.3 Member Function Documentation

10.405.3.1 `template<typename T , typename V > const V & mln::value::value_array< T, V
>::operator()(const T & v) const [inline]`

```
}
```

Access elements through a value of T. {

Definition at line 128 of file value_array.hh.

10.405.3.2 `template<typename T , typename V > const V & mln::value::value_array< T, V
>::operator[](unsigned i) const [inline]`

```
}
```

Access elements through array indexes. {

Definition at line 152 of file value_array.hh.

10.405.3.3 `template<typename T , typename V > const mln::value::set< T > &
mln::value::value_array< T, V >::vset() const [inline]`

```
}
```

Reference to the set of T.

Definition at line 144 of file value_array.hh.

10.406 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

10.406.1 Detailed Description

`template<typename E> struct mln::Vertex< E >`

[Vertex](#) category flag type.

Definition at line 53 of file vertex.hh.

10.407 mln::vertex_image< P, V, G > Class Template Reference

[Image](#) based on graph vertices.

```
#include <vertex_image.hh>
```

Inherits [image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >](#).

Public Types

- typedef [G](#) [graph_t](#)
The type of the underlying graph.
- typedef [vertex_nbh_t](#) [nbh_t](#)
Neighborhood type.
- typedef [internal::vfsite_selector< P, G >::site_function_t](#) [site_function_t](#)
Function mapping graph elements to sites.
- typedef [vertex_image< tag::psite_< P >, tag::value_< V >, tag::graph_< G > >](#) [skeleton](#)
Skeleton type.
- typedef [vertex_win_t](#) [win_t](#)
Window type.

Public Member Functions

- [vertex_image](#) ()
Constructors.
- rvalue [operator\(\)](#) (unsigned v_id) const
Value accessors/operators overloads.

10.407.1 Detailed Description

```
template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >
```

[Image](#) based on graph vertices.

Definition at line 126 of file core/image/vertex_image.hh.

10.407.2 Member Typedef Documentation

10.407.2.1 `template<typename P, typename V, typename G = util::graph> typedef G mln::vertex_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 141 of file core/image/vertex_image.hh.

10.407.2.2 `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

Definition at line 165 of file core/image/vertex_image.hh.

10.407.2.3 `template<typename P, typename V, typename G = util::graph> typedef internal::vfsite_selector<P,G>::site_function_t mln::vertex_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 145 of file core/image/vertex_image.hh.

10.407.2.4 `template<typename P, typename V, typename G = util::graph> typedef vertex_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::vertex_image< P, V, G >::skeleton`

[Skeleton](#) type.

Definition at line 152 of file core/image/vertex_image.hh.

10.407.2.5 `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

Definition at line 163 of file core/image/vertex_image.hh.

10.407.3 Constructor & Destructor Documentation

10.407.3.1 `template<typename P , typename V , typename G > mln::vertex_image< P, V, G >::vertex_image () [inline]`

Constructors.

Definition at line 247 of file core/image/vertex_image.hh.

10.407.4 Member Function Documentation

10.407.4.1 `template<typename P , typename V , typename G > vertex_image< P, V, G >::rvalue
mln::vertex_image< P, V, G >::operator() (unsigned v_id) const`

[Value](#) accessors/operators overloads.

Definition at line 292 of file core/image/vertex_image.hh.

10.408 mln::violent_cast_image< T, I > Struct Template Reference

Violently cast image values to a given type.

`#include <violent_cast_image.hh>`

Inherits `image_value_morpher< I, T, violent_cast_image< T, I > >`.

Public Types

- typedef T [lvalue](#)
Return type of read-write access.
- typedef T [rvalue](#)
Return type of read-only access.
- typedef [violent_cast_image](#)< tag::value_< T >, tag::image_< I > > [skeleton](#)
Skeleton.
- typedef T [value](#)
[Value](#) associated type.

Public Member Functions

- T [operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- T [operator\(\)](#) (const typename I::psite &p)
Mutable access is only OK for reading (not writing).
- [violent_cast_image](#) (const [Image](#)< I > &ima)
Constructor.

10.408.1 Detailed Description

`template<typename T, typename I> struct mln::violent_cast_image< T, I >`

Violently cast image values to a given type.

Definition at line 112 of file violent_cast_image.hh.

10.408.2 Member Typedef Documentation

10.408.2.1 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::lvalue`

Return type of read-write access.

Definition at line 122 of file violent_cast_image.hh.

10.408.2.2 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::rvalue`

Return type of read-only access.

Definition at line 119 of file violent_cast_image.hh.

10.408.2.3 `template<typename T, typename I> typedef violent_cast_image< tag::value_<T>, tag::image_<I> > mln::violent_cast_image< T, I >::skeleton`

Skeleton.

Definition at line 125 of file violent_cast_image.hh.

10.408.2.4 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::value`

[Value](#) associated type.

Definition at line 116 of file violent_cast_image.hh.

10.408.3 Constructor & Destructor Documentation

10.408.3.1 `template<typename T , typename I > mln::violent_cast_image< T, I >::violent_cast_image(const Image< I > & ima) [inline]`

Constructor.

Definition at line 172 of file violent_cast_image.hh.

10.408.4 Member Function Documentation

10.408.4.1 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator() (const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 191 of file violent_cast_image.hh.

10.408.4.2 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator() (const typename I::psite & p) [inline]`

Mutable access is only OK for reading (not writing).

Definition at line 200 of file violent_cast_image.hh.

10.409 mln::w_window< D, W > Struct Template Reference

Generic [w_window](#) class.

```
#include <w_window.hh>
```

Inherits [weighted_window_base< mln::window< D >, w_window< D, W > >](#).

Public Types

- typedef with_w_< [dpsites_bkd_piter](#)< [w_window](#)< D, W > >, W > [bkd_qiter](#)
Site_iterator type to browse (backward) the points of a generic [w_window](#).
- typedef D [dpsite](#)
Dpsite associated type.
- typedef with_w_< [dpsites_fwd_piter](#)< [w_window](#)< D, W > >, W > [fwd_qiter](#)
Site_iterator type to browse (forward) the points of a generic [w_window](#).
- typedef W [weight](#)
Weight associated type.

Public Member Functions

- void [clear](#) ()
Clear this window.
- [w_window](#)< D, W > & [insert](#) (const W &w, const D &d)
Insert a couple of weight w and delta-point d.
- bool [is_symmetric](#) () const
Test if the window is symmetric.
- const std::vector< D > & [std_vector](#) () const
Give access to the vector of delta-points.
- void [sym](#) ()
Apply a central symmetry to the window.
- W [w](#) (unsigned i) const
Give the i-th weight.
- [w_window](#) ()
Constructor without argument.
- const std::vector< W > & [weights](#) () const
Give access to the vector of weights.
- const [mln::window](#)< D > & [win](#) () const
Give the corresponding window.

Related Functions

(Note that these are not member functions.)

- `template<typename D, typename W>`
`std::ostream & operator<< (std::ostream &ostr, const w_window< D, W > &w_win)`
Print a weighted window `w_win` into an output stream `ostr`.
- `template<typename D, typename Wl, typename Wr>`
`bool operator== (const w_window< D, Wl > &lhs, const w_window< D, Wr > &rhs)`
Equality test between two weighted windows `lhs` and `rhs`.

10.409.1 Detailed Description

`template<typename D, typename W> struct mln::w_window< D, W >`

Generic `w_window` class. This type of `w_window` is just like a set of delta-points. The parameter `D` is the type of delta-points; the parameter `W` is the type of weights.

Definition at line 100 of file `core/w_window.hh`.

10.409.2 Member Typedef Documentation

10.409.2.1 `template<typename D, typename W> typedef with_w< dpsites_bkd_piter< w_window<D, W> >, W > mln::w_window< D, W >::bkd_qiter`

`Site_Iterator` type to browse (backward) the points of a generic `w_window`.

Definition at line 114 of file `core/w_window.hh`.

10.409.2.2 `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

Definition at line 104 of file `core/w_window.hh`.

10.409.2.3 `template<typename D, typename W> typedef with_w< dpsites_fwd_piter< w_window<D, W> >, W > mln::w_window< D, W >::fwd_qiter`

`Site_Iterator` type to browse (forward) the points of a generic `w_window`.

Definition at line 111 of file `core/w_window.hh`.

10.409.2.4 `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

Definition at line 107 of file `core/w_window.hh`.

10.409.3 Constructor & Destructor Documentation

10.409.3.1 `template<typename D , typename W > mln::w_window< D, W >::w_window ()`
`[inline]`

Constructor without argument.

Definition at line 213 of file core/w_window.hh.

10.409.4 Member Function Documentation

10.409.4.1 `template<typename D , typename W > void mln::w_window< D, W >::clear ()`
`[inline]`

Clear this window.

Definition at line 308 of file core/w_window.hh.

10.409.4.2 `template<typename D , typename W > w_window< D, W > & mln::w_window< D, W >::insert (const W & w, const D & d)` `[inline]`

Insert a couple of weight *w* and delta-point *d*.

Definition at line 254 of file core/w_window.hh.

Referenced by `mln::w_window< D, W >::sym()`, `mln::make::w_window()`, `mln::make::w_window1d()`, `mln::make::w_window3d()`, and `mln::make::w_window_directional()`.

10.409.4.3 `template<typename D , typename W > bool mln::w_window< D, W >::is_symmetric () const` `[inline]`

Test if the window is symmetric.

Definition at line 284 of file core/w_window.hh.

References `mln::w_window< D, W >::sym()`.

10.409.4.4 `template<typename D , typename W > const std::vector< D > & mln::w_window< D, W >::std_vector () const` `[inline]`

Give access to the vector of delta-points.

Definition at line 228 of file core/w_window.hh.

10.409.4.5 `template<typename D , typename W > void mln::w_window< D, W >::sym ()`
`[inline]`

Apply a central symmetry to the window.

Definition at line 296 of file core/w_window.hh.

References `mln::w_window< D, W >::insert()`.

Referenced by `mln::w_window< D, W >::is_symmetric()`.

10.409.4.6 `template<typename D , typename W > W mln::w_window< D, W >::w (unsigned i) const [inline]`

Give the `i`-th weight.

Definition at line 244 of file `core/w_window.hh`.

10.409.4.7 `template<typename D , typename W > const std::vector< W > & mln::w_window< D, W >::weights () const [inline]`

Give access to the vector of weights.

Definition at line 236 of file `core/w_window.hh`.

Referenced by `mln::w_window< D, W >::operator==()`.

10.409.4.8 `template<typename D , typename W > const mln::window< D > & mln::w_window< D, W >::win () const [inline]`

Give the corresponding window.

Definition at line 220 of file `core/w_window.hh`.

Referenced by `mln::w_window< D, W >::operator==()`.

10.409.5 Friends And Related Function Documentation

10.409.5.1 `template<typename D , typename W > std::ostream & operator<< (std::ostream & ostr, const w_window< D, W > & w_win) [related]`

Print a weighted window `w_win` into an output stream `ostr`.

Definition at line 420 of file `core/w_window.hh`.

10.409.5.2 `template<typename D , typename Wl , typename Wr > bool operator== (const w_window< D, Wl > & lhs, const w_window< D, Wr > & rhs) [related]`

Equality test between two weighted windows `lhs` and `rhs`.

Definition at line 430 of file `core/w_window.hh`.

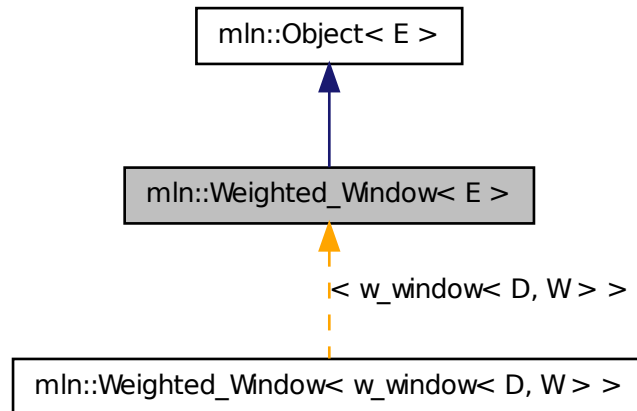
References `mln::w_window< D, W >::weights()`, and `mln::w_window< D, W >::win()`.

10.410 mln::Weighted_Window< E > Struct Template Reference

Base class for implementation classes that are `weighted_windows`.

`#include <weighted_window.hh>`

Inheritance diagram for `mln::Weighted_Window< E >`:



Related Functions

(Note that these are not member functions.)

- `template<typename W >`
`W operator- (const Weighted_Window< W > &rhs)`
Compute the symmetrical weighted window of `rhs`.

10.410.1 Detailed Description

`template<typename E> struct mln::Weighted_Window< E >`

Base class for implementation classes that are `weighted_windows`.

See also

[mln::doc::Weighted_Window](#) for a complete documentation of this class contents.

Definition at line 68 of file `weighted_window.hh`.

10.410.2 Friends And Related Function Documentation

10.410.2.1 `template<typename W > W operator- (const Weighted_Window< W > & rhs)`
`[related]`

Compute the symmetrical weighted window of `rhs`.

10.411 mln::win::backdiag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <backdiag2d.hh>
```

Inherits `classical_window_base< dpoint2d, backdiag2d >`.

Public Member Functions

- [backdiag2d](#) (unsigned length)

Constructor.

- unsigned [length](#) () const

Give the diagonal length, that is, its width.

10.411.1 Detailed Description

Diagonal line window defined on the 2D square grid. An [backdiag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```
*  ○
*   ○
*    ×
*     ○
*      ○
*
*
```

is defined with `length = 5`.

Definition at line 63 of file `backdiag2d.hh`.

10.411.2 Constructor & Destructor Documentation

10.411.2.1 mln::win::backdiag2d::backdiag2d (unsigned length) [inline]

Constructor.

Parameters

[in] **length** Length, thus width, of the diagonal line.

Precondition

`length` is odd.

Definition at line 93 of file `backdiag2d.hh`.

10.411.3 Member Function Documentation

10.411.3.1 unsigned mln::win::backdiag2d::length () const [inline]

Give the diagonal length, that is, its width.

Definition at line 105 of file backdiag2d.hh.

10.412 mln::win::ball< G, C > Struct Template Reference

Generic ball window defined on a given grid.

```
#include <ball.hh>
```

Inherits classical_window_base< dpoint< G, C >, ball< G, C > >.

Public Member Functions

- [ball](#) (unsigned diameter)
Constructor.
- unsigned [diameter](#) () const
Give the ball diameter.

10.412.1 Detailed Description

```
template<typename G, typename C> struct mln::win::ball< G, C >
```

Generic ball window defined on a given grid. A ball is centered and symmetric; so its diameter is odd.

G is the given grid on which the ball is defined and C is the type of coordinates.

Definition at line 71 of file ball.hh.

10.412.2 Constructor & Destructor Documentation

10.412.2.1 template<typename G , typename C > mln::win::ball< G, C >::ball (unsigned diameter) [inline]

Constructor.

Parameters

[in] *diameter* Diameter of the ball.

Precondition

diameter is odd.

Definition at line 99 of file ball.hh.

References mln::literal::origin.

10.412.3 Member Function Documentation

10.412.3.1 `template<typename G , typename C > unsigned mln::win::ball< G, C >::diameter () const [inline]`

Give the ball diameter.

Definition at line 122 of file ball.hh.

10.413 mln::win::cube3d Struct Reference

Cube window defined on the 3D grid.

```
#include <cube3d.hh>
```

Inherits `classical_window_base< dpoint3d, cube3d >`.

Public Member Functions

- `cube3d` (unsigned length)
Constructor.
- unsigned `length` () const
Give the cube length, that is, its height.

10.413.1 Detailed Description

Cube window defined on the 3D grid. An `cube3d` is centered and symmetric; so its height (length) is odd. For instance:

```
*   o o o
*   o o o
*   o o o

*   o o o
*   o x o
*   o o o

*   o o o
*   o o o
*   o o o
*
```

is defined with `length = 3`.

Definition at line 69 of file cube3d.hh.

10.413.2 Constructor & Destructor Documentation

10.413.2.1 `mln::win::cube3d::cube3d (unsigned length) [inline]`

Constructor.

Parameters

[in] *length* Length, thus height, of the [cube3d](#).

Precondition

length is odd.

Definition at line 99 of file `cube3d.hh`.

10.413.3 Member Function Documentation**10.413.3.1 unsigned [mln::win::cube3d::length](#) () const [inline]**

Give the cube length, that is, its height.

Definition at line 113 of file `cube3d.hh`.

10.414 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square grid.

```
#include <cuboid3d.hh>
```

Inherits `classical_window_base< dpoint3d, cuboid3d >`.

Public Member Functions

- [cuboid3d](#) (unsigned depth, unsigned height, unsigned width)

Constructor.

- unsigned [volume](#) () const

Return the volume of the cuboid.

- unsigned [depth](#) () const

Accessors.

- unsigned [height](#) () const

Return the height of the cuboid.

- unsigned [width](#) () const

Return the width of the cuboid.

10.414.1 Detailed Description

Cuboid defined on the 3-D square grid. A [cuboid3d](#) is a 3-D window with cuboid (also known as rectangular prism or rectangular parallelepiped) shape. It is centered and symmetric.

For instance:

```

      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o

      o o o o o o o
      o o o o o o o
      o o o x o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o

      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o
      o o o o o o o

```

is defined with depth = 3, height = 5 and width = 7.

Reference: <http://en.wikipedia.org/wiki/Cuboid>

Definition at line 80 of file cuboid3d.hh.

10.414.2 Constructor & Destructor Documentation

10.414.2.1 mln::win::cuboid3d::cuboid3d (unsigned *depth*, unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters

- [in] *depth* The depth of the [cuboid3d](#).
- [in] *height* The height of the [cuboid3d](#).
- [in] *width* The width of the [cuboid3d](#).

Precondition

Argument *depth*, *height* and *width* must be odd.

Definition at line 125 of file cuboid3d.hh.

10.414.3 Member Function Documentation

10.414.3.1 unsigned mln::win::cuboid3d::depth () const [inline]

Accessors.

Return the depth of the cuboid.

Definition at line 146 of file cuboid3d.hh.

10.414.3.2 unsigned mln::win::cuboid3d::height () const [inline]

Return the height of the cuboid.

Definition at line 153 of file cuboid3d.hh.

10.414.3.3 unsigned mln::win::cuboid3d::volume () const [inline]

Return the volume of the cuboid.

Definition at line 167 of file cuboid3d.hh.

10.414.3.4 unsigned mln::win::cuboid3d::width () const [inline]

Return the width of the cuboid.

Definition at line 160 of file cuboid3d.hh.

10.415 mln::win::diag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <diag2d.hh>
```

Inherits classical_window_base< dpoint2d, diag2d >.

Public Member Functions

- [diag2d](#) (unsigned length)
Constructor.
- unsigned [length](#) () const
Give the diagonal length, that is, its width.

10.415.1 Detailed Description

Diagonal line window defined on the 2D square grid. An [diag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```

*           o
*           o
*        x
*     o
*  o
*

```

is defined with length = 5.

Definition at line 63 of file diag2d.hh.

10.415.2 Constructor & Destructor Documentation

10.415.2.1 mln::win::diag2d::diag2d (unsigned length) [inline]

Constructor.

Parameters

[in] *length* Length, thus width, of the diagonal line.

Precondition

`length` is odd.

Definition at line 93 of file diag2d.hh.

10.415.3 Member Function Documentation**10.415.3.1 unsigned mln::win::diag2d::length () const [inline]**

Give the diagonal length, that is, its width.

Definition at line 106 of file diag2d.hh.

10.416 mln::win::line< M, i, C > Struct Template Reference

Generic line window defined on a given grid in the given dimension.

```
#include <line.hh>
```

Inherits classical_window_base< dpoint< M, C >, line< M, i, C > >.

Public Types

- enum
Direction.

Public Member Functions

- unsigned `length` () const
Give the line length.
- `line` (unsigned length)
Constructor.
- unsigned `size` () const
Give the line size, that is, its length.

10.416.1 Detailed Description

```
template<typename M, unsigned i, typename C> struct mln::win::line< M, i, C >
```

Generic line window defined on a given grid in the given dimension. An line is centered and symmetric; so its length is odd.

M is the given grid on which the line is defined, i is the given dimension of the line end C is the type of the coordinates.

See also

mln::win::hline2d for an exemple of his use.

Definition at line 73 of file win/line.hh.

10.416.2 Member Enumeration Documentation

10.416.2.1 `template<typename M , unsigned i, typename C > anonymous enum`

Direction.

Definition at line 76 of file win/line.hh.

10.416.3 Constructor & Destructor Documentation

10.416.3.1 `template<typename M , unsigned i, typename C > mln::win::line< M, i, C >::line (unsigned length) [inline]`

Constructor.

Parameters

[in] *length* Length of the line.

Precondition

length is odd.

Definition at line 106 of file win/line.hh.

References mln::dpoint< G, C >::set_all().

10.416.4 Member Function Documentation

10.416.4.1 `template<typename M , unsigned i, typename C > unsigned mln::win::line< M, i, C >::length () const [inline]`

Give the line length.

Definition at line 125 of file win/line.hh.

10.416.4.2 `template<typename M , unsigned i, typename C > unsigned mln::win::line< M, i, C >::size () const [inline]`

Give the line size, that is, its length.

Definition at line 132 of file win/line.hh.

10.417 mln::win::multiple< W, F > Class Template Reference

Multiple window.

```
#include <multiple.hh>
```

Inherits window_base< W::dpsite, multiple< W, F > >.

10.417.1 Detailed Description

```
template<typename W, typename F> class mln::win::multiple< W, F >
```

Multiple window.

Definition at line 76 of file multiple.hh.

10.418 mln::win::multiple_size< n, W, F > Class Template Reference

Definition of a multiple-size window.

```
#include <multiple_size.hh>
```

Inherits window_base< W::dpsite, multiple_size< n, W, F > >.

10.418.1 Detailed Description

```
template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >
```

Definition of a multiple-size window.

Definition at line 78 of file multiple_size.hh.

10.419 mln::win::octagon2d Struct Reference

Octagon window defined on the 2D square grid.

```
#include <octagon2d.hh>
```

Inherits classical_window_base< dpoint2d, octagon2d >.

Public Member Functions

- unsigned [area](#) () const
Give the area.
- unsigned [length](#) () const
Give the octagon length, that is, its width.
- [octagon2d](#) (unsigned length)
Constructor.

10.419.1 Detailed Description

Octagon window defined on the 2D square grid. An [octagon2d](#) is centered and symmetric.

The length L of the octagon is such as $L = 6 * l + 1$ where $l \geq 0$.

For instance:

```

*      o o o
*    o o o o o
*  o o o o o o o
* o o o x o o o
* o o o o o o o
*   o o o o o
*     o o o
*
```

is defined with $L = 7$ ($l = 1$).

Definition at line 67 of file octagon2d.hh.

10.419.2 Constructor & Destructor Documentation

10.419.2.1 `mln::win::octagon2d::octagon2d (unsigned length) [inline]`

Constructor.

Parameters

[in] *length* Length, of the octagon.

Precondition

length is such as $length = 6 * x + 1$ where $x \geq 0$.

Definition at line 101 of file octagon2d.hh.

10.419.3 Member Function Documentation

10.419.3.1 `unsigned mln::win::octagon2d::area () const [inline]`

Give the area.

Definition at line 157 of file octagon2d.hh.

10.419.3.2 `unsigned mln::win::octagon2d::length () const [inline]`

Give the octagon length, that is, its width.

Definition at line 145 of file octagon2d.hh.

10.420 mln::win::rectangle2d Struct Reference

Rectangular window defined on the 2D square grid.

```
#include <rectangle2d.hh>
```

Inherits `classical_window_base< dpoint2d, rectangle2d >`.

Public Member Functions

- unsigned `area` () const
Give the rectangle area.
- unsigned `height` () const
Give the rectangle height.
- `rectangle2d` (unsigned height, unsigned width)
Constructor.
- const std::vector< `dpoint2d` > & `std_vector` () const
Give the std vector of delta-points.
- unsigned `width` () const
Give the rectangle width.

10.420.1 Detailed Description

Rectangular window defined on the 2D square grid. A `rectangle2d` is a 2D window with rectangular shape. It is centered and symmetric.

For instance:

```
*  o o o o o
*  o o x o o
*  o o o o o
*
```

is defined with height = 3 and width = 5.

Definition at line 64 of file `rectangle2d.hh`.

10.420.2 Constructor & Destructor Documentation

10.420.2.1 mln::win::rectangle2d::rectangle2d (unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters

- [in] *height* Height of the `rectangle2d`.
- [in] *width* Width of the `rectangle2d`.

Precondition

Height and width are odd.

Definition at line 106 of file rectangle2d.hh.

10.420.3 Member Function Documentation**10.420.3.1 unsigned mln::win::rectangle2d::area () const [inline]**

Give the rectangle area.

Definition at line 132 of file rectangle2d.hh.

10.420.3.2 unsigned mln::win::rectangle2d::height () const [inline]

Give the rectangle height.

Definition at line 120 of file rectangle2d.hh.

10.420.3.3 const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector () const [inline]

Give the std vector of delta-points.

Definition at line 145 of file rectangle2d.hh.

10.420.3.4 unsigned mln::win::rectangle2d::width () const [inline]

Give the rectangle width.

Definition at line 126 of file rectangle2d.hh.

10.421 mln::Window< E > Struct Template Reference

Base class for implementation classes that are windows.

```
#include <window.hh>
```

```
template<typename E> struct mln::Window< E >
```

See also

[mnl::doc::Window](#) for a complete documentation of this class contents.

Definition at line 87 of file concept/window.hh.

Generic window class.

```
#include <window.hh>
```

Inherits window base< D, window< D > >.

- typedef `dpsites_bkd_piter` < `window` < D > > `bkd_qiter`
Site_Iterator type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.
- typedef `dpsites_fwd_piter` < `window` < D > > `fwd_qiter`
Site_Iterator type to browse the points of a basic window w.r.t. the ordering of delta-points.
- typedef `fwd_qiter` `qiter`
Site_Iterator type to browse the points of a basic window whatever the ordering of delta-points.
- typedef `window` < D > `regular`
Regular window associated type.

Public Member Functions

- void [clear](#) ()
Clear the window.
- unsigned [delta](#) () const
Give the maximum coordinate gap between the window center and a window point.
- const D & [dp](#) (unsigned i) const
Give the i -th delta-point.
- bool [has](#) (const D &dp) const
Test if dp is in this window definition.
- [window](#)< D > & [insert](#) (const D &dp)
Insert a delta-point dp .
- template<typename W >
[window](#)< D > & [insert](#) (const [Window](#)< W > &win)
Insert another window win .
- bool [is_centered](#) () const
Test if the window is centered.
- bool [is_empty](#) () const
Test if the window is empty (null size; no delta-point).
- bool [is_symmetric](#) () const
- void [print](#) (std::ostream &ostr) const
Print the window definition into $ostr$.
- unsigned [size](#) () const
Give the window size, i.e., the number of delta-sites.
- const std::vector< D > & [std_vector](#) () const
Give the std vector of delta-points.
- void [sym](#) ()
Apply a central symmetry to the target window.
- [window](#) ()
Constructor without argument.
- [window](#)< D > & [insert](#) (const typename D::coord &dind)

Related Functions

(Note that these are not member functions.)

- `template<typename D >`
`bool operator== (const window< D > &lhs, const window< D > &rhs)`
Equality comparison between windows lhs and rhs.

10.422.1 Detailed Description

`template<typename D> class mln::window< D >`

Generic window class. This type of window is just like a set of delta-points. The parameter is D, type of delta-point.

Definition at line 85 of file window.hh.

10.422.2 Member Typedef Documentation

10.422.2.1 `template<typename D> typedef dpsites_bkd_piter< window<D> > mln::window< D >::bkd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.

Definition at line 124 of file window.hh.

10.422.2.2 `template<typename D> typedef dpsites_fwd_piter< window<D> > mln::window< D >::fwd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the ordering of delta-points.

Definition at line 119 of file window.hh.

10.422.2.3 `template<typename D> typedef fwd_qiter mln::window< D >::qiter`

[Site_Iterator](#) type to browse the points of a basic window whatever the ordering of delta-points.

Definition at line 129 of file window.hh.

10.422.2.4 `template<typename D> typedef window<D> mln::window< D >::regular`

Regular window associated type.

Definition at line 90 of file window.hh.

10.422.3 Constructor & Destructor Documentation

10.422.3.1 `template<typename D > mln::window< D >::window () [inline]`

Constructor without argument.

The constructed window is empty.

Definition at line 207 of file window.hh.

10.422.4 Member Function Documentation

10.422.4.1 `template<typename D> void mln::window<D>::clear () [inline]`

Clear the window.

Definition at line 254 of file window.hh.

10.422.4.2 `template<typename D> unsigned mln::window<D>::delta () const [inline]`

Give the maximum coordinate gap between the window center and a window point.

Definition at line 262 of file window.hh.

References `mln::window<D>::dp()`, and `mln::window<D>::size()`.

10.422.4.3 `template<typename D> const D & mln::window<D>::dp (unsigned i) const [inline]`

Give the *i*-th delta-point.

Definition at line 302 of file window.hh.

References `mln::window<D>::size()`.

Referenced by `mln::window<D>::delta()`, and `mln::window<D>::insert()`.

10.422.4.4 `template<typename D> bool mln::window<D>::has (const D & dp) const [inline]`

Test if `dp` is in this window definition.

Definition at line 311 of file window.hh.

10.422.4.5 `template<typename D> window<D> & mln::window<D>::insert (const D & dp) [inline]`

Insert a delta-point `dp`.

Definition at line 327 of file window.hh.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c2_3d_sli()`, `mln::c4_3d()`, `mln::c6()`, `mln::window<D>::insert()`, `mln::morpho::line_gradient()`, `mln::window<D>::sym()`, `mln::convert::to_upper_window()`, `mln::convert::to_window()`, `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

10.422.4.6 `template<typename D> template<typename W> window<D> & mln::window<D>::insert (const Window<W> & win) [inline]`

Insert another window `win`.

Definition at line 337 of file window.hh.

10.422.4.7 **template<typename D > window< D > & mln::window< D >::insert (const typename D::coord & *dind*) [inline]**

Insertion of a delta-point with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 349 of file window.hh.

References mln::window< D >::dp(), and mln::window< D >::insert().

10.422.4.8 **template<typename D > bool mln::window< D >::is_centered () const [inline]**

Test if the window is centered.

Returns

True if the delta-point 0 belongs to the window.

Definition at line 226 of file window.hh.

References mln::literal::zero.

10.422.4.9 **template<typename D > bool mln::window< D >::is_empty () const [inline]**

Test if the window is empty (null size; no delta-point).

Definition at line 246 of file window.hh.

10.422.4.10 **template<typename D > bool mln::window< D >::is_symmetric () const [inline]**

Test if the window is symmetric.

Returns

True if for every dp of this window, -dp is also in this window.

Definition at line 216 of file window.hh.

References mln::window< D >::sym().

10.422.4.11 **template<typename D > void mln::window< D >::print (std::ostream & *ostr*) const [inline]**

Print the window definition into *ostr*.

Definition at line 390 of file window.hh.

10.422.4.12 **template<typename D > unsigned mln::window< D >::size () const [inline]**

Give the window size, i.e., the number of delta-sites.

Definition at line 294 of file window.hh.

Referenced by mln::window< D >::delta(), mln::window< D >::dp(), mln::window< D >::sym(), mln::win_c4p(), mln::win_c4p_3d(), mln::win_c8p(), and mln::win_c8p_3d().

10.422.4.13 `template<typename D> const std::vector< D> & mln::window< D>::std_vector () const [inline]`

Give the std vector of delta-points.

Definition at line 319 of file window.hh.

10.422.4.14 `template<typename D> void mln::window< D>::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 234 of file window.hh.

References `mln::window< D>::insert()`, and `mln::window< D>::size()`.

Referenced by `mln::window< D>::is_symmetric()`.

10.422.5 Friends And Related Function Documentation

10.422.5.1 `template<typename D> bool operator== (const window< D> & lhs, const window< D> & rhs) [related]`

Equality comparison between windows `lhs` and `rhs`.

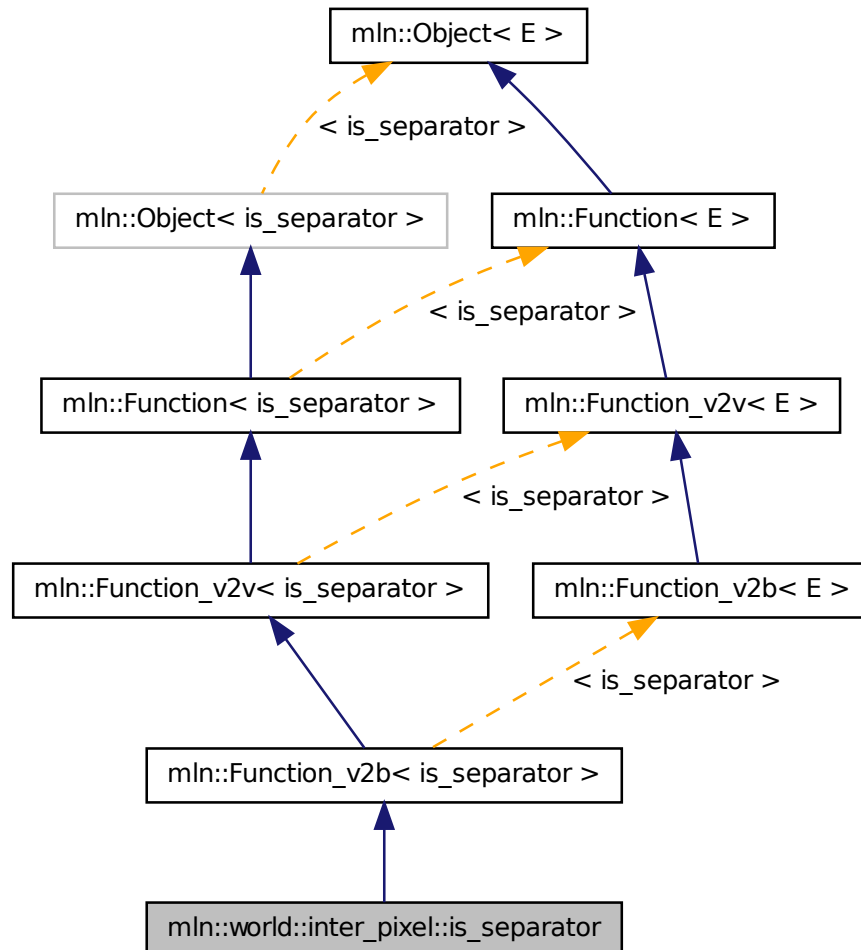
Definition at line 400 of file window.hh.

10.423 mln::world::inter_pixel::is_separator Struct Reference

Functor returning whether a site is a separator in an inter-pixel image.

```
#include <is_separator.hh>
```

Inheritance diagram for mln::world::inter_pixel::is_separator:



10.423.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

Definition at line 52 of file is_separator.hh.

10.424 trait::graph< I > Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```

10.424.1 Detailed Description

template<typename I> struct trait::graph< I >

Graph traits.

Definition at line 73 of file morpho.hh.

10.425 **trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference**

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

10.425.1 Detailed Description

template<typename G, typename V> struct trait::graph< mln::complex_image< 1, G, V > >

Graph traits for 1-complexes images.

Definition at line 133 of file morpho.hh.

10.426 **trait::graph< mln::image2d< T > > Struct Template Reference**

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

10.426.1 Detailed Description

template<typename T> struct trait::graph< mln::image2d< T > >

Graph traits for [mln::image2d](#).

Definition at line 94 of file morpho.hh.

Index

- ~proxy
 - mln::value::proxy, [1116](#)
- ~soft_heap
 - mln::util::soft_heap, [1073](#)
- ~tracked_ptr
 - mln::util::tracked_ptr, [1076](#)
- _1
 - mln::algebra::h_mat, [542](#)
- 1D neighborhoods, [101](#)
- 1D windows, [112](#)
- 2D neighborhoods, [102](#)
- 2D windows, [113](#)
- 3D neighborhoods, [104](#)
- 3D windows, [115](#)
- a_point_of
 - mln, [148](#)
- abs
 - mln::data, [204](#)
 - mln::math, [350](#), [351](#)
- abs_inplace
 - mln::data, [204](#)
- Accumulators, [99](#)
- add
 - mln::topo::n_faces_set, [1008](#)
- add_child
 - mln::util::tree_node, [1081](#)
- add_edge
 - mln::util::graph, [1045](#)
- add_face
 - mln::topo::complex, [988](#)
- add_location
 - mln::geom::complex_geometry, [721](#)
- add_tree_down
 - mln::util::tree, [1078](#)
- add_tree_up
 - mln::util::tree, [1078](#)
- add_vertex
 - mln::util::graph, [1045](#)
- add_vertices
 - mln::util::graph, [1046](#)
- addr
 - mln::topo::complex, [988](#)
- adj_higher_dim_connected_n_face_bkd_iter
 - mln::topo::adj_higher_dim_connected_n_face_bkd_iter, [959](#)
- adj_higher_dim_connected_n_face_fwd_iter
 - mln::topo::adj_higher_dim_connected_n_face_fwd_iter, [960](#)
- adj_higher_face_bkd_iter
 - mln::topo::adj_higher_face_bkd_iter, [962](#)
- adj_higher_face_fwd_iter
 - mln::topo::adj_higher_face_fwd_iter, [963](#)
- adj_lower_dim_connected_n_face_bkd_iter
 - mln::topo::adj_lower_dim_connected_n_face_bkd_iter, [964](#)
- adj_lower_dim_connected_n_face_fwd_iter
 - mln::topo::adj_lower_dim_connected_n_face_fwd_iter, [965](#)
- adj_lower_face_bkd_iter
 - mln::topo::adj_lower_face_bkd_iter, [966](#)
- adj_lower_face_fwd_iter
 - mln::topo::adj_lower_face_fwd_iter, [967](#)
- adj_lower_higher_face_bkd_iter
 - mln::topo::adj_lower_higher_face_bkd_iter, [968](#)
- adj_lower_higher_face_fwd_iter
 - mln::topo::adj_lower_higher_face_fwd_iter, [969](#)
- adj_m_face_bkd_iter
 - mln::topo::adj_m_face_bkd_iter, [971](#)
- adj_m_face_fwd_iter
 - mln::topo::adj_m_face_fwd_iter, [972](#)
- adjacency_matrix
 - mln::util::adjacency_matrix, [1019](#)
- adjust
 - mln::border, [188](#)
 - mln::extension, [238](#), [239](#)
- adjust_duplicate
 - mln::extension, [239](#)
- adjust_fill
 - mln::extension, [239](#)
- algebraic_face
 - mln::topo::algebraic_face, [975](#)
- algebraic_n_face
 - mln::topo::algebraic_n_face, [980](#)
- and_inplace
 - mln::logical, [324](#)
- and_not

- mln::logical, [324](#)
- and_not_inplace
 - mln::logical, [325](#)
- antialiased
 - mln::subsampling, [392](#)
- apex
 - mln::util::branch, [1028](#)
- append
 - mln::p_array, [842](#)
 - mln::util::array, [1023](#)
- apply
 - mln::data, [205](#)
- apply_p2p
 - mln, [149](#)
- area
 - mln::accu::site_set::rectangularity, [506](#)
 - mln::morpho::attribute::sharpness, [830](#)
 - mln::morpho::attribute::volume, [834](#)
 - mln::win::octagon2d, [1146](#)
 - mln::win::rectangle2d, [1148](#)
- argument
 - mln::accu::shape::height, [502](#)
 - mln::accu::shape::volume, [504](#)
 - mln::doc::Accumulator, [596](#)
- array
 - mln::util::array, [1023](#)
- at
 - mln::opt, [384](#)
- attachment
 - mln::make, [332](#)
- backdiag2d
 - mln::win::backdiag2d, [1137](#)
- background
 - mln::labeling, [296](#)
- ball
 - mln::win::ball, [1138](#)
- base_level
 - mln::morpho::attribute::height, [828](#)
- Basic types, [95](#), [108](#)
- bbox
 - mln::accu::site_set::rectangularity, [506](#)
 - mln::Box, [559](#)
 - mln::box, [553](#)
 - mln::doc::Box, [598](#)
 - mln::doc::Fastest_Image, [606](#)
 - mln::doc::Image, [615](#)
 - mln::geom, [253](#), [254](#)
 - mln::image1d, [762](#)
 - mln::image2d, [767](#)
 - mln::image3d, [775](#)
 - mln::labeled_image, [786](#)
 - mln::labeled_image_base, [789](#)
 - mln::p_line2d, [881](#)
 - mln::p_run, [907](#)
- bbox_t
 - mln::labeled_image, [785](#)
 - mln::labeled_image_base, [789](#)
- bboxes
 - mln::labeled_image, [786](#)
 - mln::labeled_image_base, [789](#)
- before
 - mln, [162](#)
- begin
 - mln::p_line2d, [881](#)
- bin_1complex_image2d
 - mln, [144](#)
- bin_2complex_image3df
 - mln, [144](#)
- binarization
 - mln::binarization, [187](#)
- bkd_citer
 - mln::topo::complex, [987](#)
- bkd_eiter
 - mln::util::array, [1022](#)
 - mln::util::set, [1066](#)
- bkd_niter
 - mln::doc::Neighborhood, [619](#)
 - mln::graph_elt_mixed_neighborhood, [729](#)
 - mln::graph_elt_neighborhood, [735](#)
 - mln::graph_elt_neighborhood_if, [736](#)
 - mln::neighb, [835](#)
- bkd_piter
 - mln::box, [551](#)
 - mln::doc::Box, [597](#)
 - mln::doc::Fastest_Image, [604](#)
 - mln::doc::Image, [613](#)
 - mln::doc::Site_Set, [629](#)
 - mln::hexa, [755](#)
 - mln::image2d_h, [770](#)
 - mln::p_array, [841](#)
 - mln::p_centered, [846](#)
 - mln::p_complex, [849](#)
 - mln::p_edges, [854](#)
 - mln::p_faces, [859](#)
 - mln::p_if, [864](#)
 - mln::p_image, [867](#)
 - mln::p_key, [875](#)
 - mln::p_line2d, [880](#)
 - mln::p_mutable_array_of, [884](#)
 - mln::p_priority, [890](#)
 - mln::p_queue, [896](#)
 - mln::p_queue_fast, [901](#)
 - mln::p_run, [906](#)
 - mln::p_set, [911](#)
 - mln::p_transformed, [915](#)
 - mln::p_vaccess, [919](#)
 - mln::p_vertices, [924](#)

- bkd_pixter1d
 - mln::bkd_pixter1d, 545
- bkd_pixter2d
 - mln::bkd_pixter2d, 546
- bkd_pixter3d
 - mln::bkd_pixter3d, 548
- bkd_qiter
 - mln::doc::Weighted_Window, 635
 - mln::doc::Window, 637
 - mln::graph_elt_mixed_window, 731
 - mln::graph_elt_window, 739
 - mln::graph_elt_window_if, 743
 - mln::w_window, 1133
 - mln::window, 1151
- bkd_viter
 - mln::doc::Value_Set, 633
 - mln::value::lut_vec, 1112
- black
 - mln::literal, 320
- blobs
 - mln::canvas::labeling, 195
 - mln::labeling, 296
- blobs_and_compute
 - mln::labeling, 296
- blue
 - mln::literal, 320
- border
 - mln::doc::Fastest_Image, 606
 - mln::image1d, 762
 - mln::image2d, 767
 - mln::image3d, 775
- box
 - mln::box, 552
 - mln::draw, 232
- box1d
 - mln, 144
 - mln::make, 332
- box2d
 - mln, 144
 - mln::make, 333
- box2d_h
 - mln, 145
 - mln::make, 334
- box3d
 - mln, 145
 - mln::make, 334, 335
- box_plain
 - mln::draw, 233
- box_runstart_piter
 - mln::box_runstart_piter, 563
- branch
 - mln::util::branch, 1027
- brown
 - mln::literal, 320
- buffer
 - mln::doc::Fastest_Image, 606
 - mln::image1d, 762
 - mln::image2d, 767
 - mln::image3d, 775
- c18
 - modneighb3d, 105
- c2
 - modneighb1d, 102
- c26
 - modneighb3d, 105
- c2_3d_sli
 - modneighb3d, 106
- c2_col
 - modneighb2d, 103
- c2_row
 - modneighb2d, 103
- c4
 - modneighb2d, 103
- c4_3d
 - modneighb3d, 106
- c6
 - modneighb3d, 107
- c8
 - modneighb2d, 104
- c8_3d
 - modneighb3d, 107
- can_stop
 - mln::accu::logic::land_basic, 446
 - mln::accu::logic::lor_basic, 449
- Canvas, 99
- card
 - mln::set, 389
- cast
 - mln::value, 424
- Category
 - mln::util::vertex, 1086
- category
 - mln::util::edge, 1036
- cell
 - mln::make, 335
- center
 - mln::p_centered, 847
- center_only_iter
 - mln::topo::center_only_iter, 983
- center_t
 - mln::graph_elt_mixed_window, 731
 - mln::graph_elt_window, 739
 - mln::graph_window_piter, 751
- center_val
 - mln::dpoints_bkd_pixter, 645
 - mln::dpoints_fwd_pixter, 648
- centered_bkd_iter_adapter

- mln::topo::centered_bkd_iter_adapter, 984
- centered_fwd_iter_adapter
 - mln::topo::centered_fwd_iter_adapter, 985
- chamfer
 - mln::geom, 254
- change
 - mln::p_array, 843
- change_both
 - mln::util::couple, 1033
 - mln::util::ord_pair, 1061
- change_extension
 - mln::extension_val, 662
- change_first
 - mln::util::couple, 1033
 - mln::util::ord_pair, 1061
- change_graph
 - mln::util::edge, 1036
 - mln::util::vertex, 1087
- change_key
 - mln::p_key, 876
- change_keys
 - mln::p_key, 876
- change_mask
 - mln::graph_elt_window_if, 745
- change_second
 - mln::util::couple, 1033
 - mln::util::ord_pair, 1061
- change_target
 - mln::complex_psite, 586
 - mln::p_transformed_piter, 917
- change_target_site_set
 - mln::graph_window_piter, 752
- change_to
 - mln::pixel, 929
- check_consistency
 - mln::util::tree, 1078
 - mln::util::tree_node, 1081
- children
 - mln::util::tree_node, 1082
- clear
 - mln::p_array, 843
 - mln::p_image, 868
 - mln::p_key, 876
 - mln::p_mutable_array_of, 885
 - mln::p_priority, 891
 - mln::p_queue, 897
 - mln::p_queue_fast, 902
 - mln::p_set, 912
 - mln::util::array, 1023
 - mln::util::fibonacci_heap, 1040
 - mln::util::set, 1066
 - mln::util::soft_heap, 1073
 - mln::w_window, 1134
 - mln::window, 1152
- closing
 - mln::morpho::elementary, 365
- colorize
 - mln::labeling, 297, 298
- complementation
 - mln::morpho, 356
- complementation_inplace
 - mln::morpho, 356
- complex
 - mln::topo::complex, 987
- Complex based, 109
- complex_geometry
 - mln::geom::complex_geometry, 721
- complex_image
 - mln::complex_image, 581
- complex_neighborhood_bkd_piter
 - mln::complex_neighborhood_bkd_piter, 583
- complex_neighborhood_fwd_piter
 - mln::complex_neighborhood_fwd_piter, 584
- complex_psite
 - mln::complex_psite, 586
- complex_window_bkd_piter
 - mln::complex_window_bkd_piter, 589
- complex_window_fwd_piter
 - mln::complex_window_fwd_piter, 591
- compose
 - mln, 149
- composed
 - mln::fun::x2x::composed, 700
- compute
 - mln::accu, 164
 - mln::data, 205
 - mln::graph, 264
 - mln::histo, 266
 - mln::labeling, 298, 299
 - mln::labeling::impl::generic, 310, 311
 - mln::set, 389
- compute_attribute_image
 - mln::morpho::tree, 370
- compute_attribute_image_from
 - mln::morpho::tree, 370
- compute_fastest
 - mln::labeling::impl, 308, 309
- compute_has
 - mln::p_queue_fast, 902
- compute_image
 - mln::labeling, 300
- compute_parent
 - mln::morpho::tree, 371
- compute_with_weights
 - mln::set, 390
- contrast
 - mln::morpho, 356
- convert

- mln::data, 206
- convolve
 - mln::linear::local, 316
- coord
 - mln::def, 229
 - mln::doc::Dpoint, 600
 - mln::doc::Fastest_Image, 604
 - mln::doc::Image, 613
 - mln::doc::Point_Site, 624
 - mln::dpoint, 641
 - mln::point, 937
- coordf
 - mln::def, 229
- count
 - mln::accu::stat::mean, 515
- couple
 - mln::make, 336
- cplx
 - mln::p_complex, 850, 851
 - mln::p_faces, 860
 - mln::topo::algebraic_face, 975
 - mln::topo::algebraic_n_face, 980
 - mln::topo::face, 991
 - mln::topo::n_face, 1003
- crop_wrt
 - mln::box, 553
- cube3d
 - mln::win::cube3d, 1139
- cuboid3d
 - mln::win::cuboid3d, 1141
- cyan
 - mln::literal, 320
- D
 - mln::topo::is_simple_cell, 1001
- dark_gray
 - mln::literal, 320
- dashed_line
 - mln::draw, 233
- data
 - mln::topo::algebraic_face, 975
 - mln::topo::algebraic_n_face, 980
 - mln::topo::face, 991
 - mln::topo::n_face, 1003
- data_t
 - mln::fun::x2x::rotation, 703
 - mln::fun::x2x::translation, 706
- dec_face_id
 - mln::topo::algebraic_face, 975
 - mln::topo::algebraic_n_face, 980
 - mln::topo::face, 991
 - mln::topo::n_face, 1003
- dec_n
 - mln::topo::algebraic_face, 976
 - mln::topo::face, 992
- decorated_image
 - mln::decorated_image, 593
- decoration
 - mln::decorated_image, 593
- deepness
 - mln::util::branch_iter, 1029
 - mln::util::branch_iter_ind, 1030
- delete_tree_node
 - mln::util::tree_node, 1082
- delta
 - mln::doc::Weighted_Window, 636
 - mln::geom, 254
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_elt_window_if, 745
 - mln::graph_window_base, 747
 - mln::point, 937
 - mln::window, 1152
- delta_index
 - mln::doc::Fastest_Image, 606
 - mln::image1d, 762
 - mln::image2d, 767
 - mln::image3d, 775
- depth
 - mln::win::cuboid3d, 1141
- detach
 - mln::topo, 399
- detachment
 - mln::make, 336
- diag2d
 - mln::win::diag2d, 1142
- diameter
 - mln::win::ball, 1139
- diff
 - mln::Box, 560
 - mln::Site_Set, 952
 - mln::win, 431
- diff_abs
 - mln::arith, 176
- dilation
 - mln::morpho, 356
- dim
 - mln::complex_image, 581
 - mln::doc::Dpoint, 600
 - mln::doc::Point_Site, 625
 - mln::dpoint, 642
 - mln::point, 938
- direct
 - mln::morpho::tree::filter, 376
- discrete_plane_1complex_geometry
 - mln, 145
- discrete_plane_2complex_geometry
 - mln, 145

- disk2d
 - modwin2d, 114
- display_branch
 - mln::util, 414
- display_tree
 - mln::util, 414
- distance_and_closest_point_geodesic
 - mln::transform, 407
- distance_and_influence_zone_geodesic
 - mln::transform, 408
- distance_front
 - mln::canvas, 193
 - mln::transform, 408
- distance_geodesic
 - mln::canvas, 193
 - mln::transform, 408
- div
 - mln::arith, 176
- div_cst
 - mln::arith, 177
- div_inplace
 - mln::arith, 177
- domain
 - mln::complex_image, 581
 - mln::doc::Fastest_Image, 606
 - mln::doc::Image, 615
 - mln::extended, 656
 - mln::flat_image, 665
 - mln::hexa, 756
 - mln::image1d, 762
 - mln::image2d, 767
 - mln::image2d_h, 771
 - mln::image3d, 776
 - mln::lazy_image, 792
 - mln::p2p_image, 839
 - mln::sub_image, 955
 - mln::sub_image_if, 956
 - mln::tr_image, 1014
 - mln::transformed_image, 1016
 - mln::unproject_image, 1018
- Domain morphers, 97
- domain_t
 - mln::value::stack_image, 1124
- dp
 - mln::window, 1152
- dpoint
 - mln::doc::Dpoint, 600
 - mln::doc::Fastest_Image, 604
 - mln::doc::Image, 613
 - mln::doc::Neighborhood, 619
 - mln::doc::Point_Site, 624
 - mln::doc::Weighted_Window, 635
 - mln::dpoint, 642
- dpoint1d
 - mln, 145
- dpoint2d
 - mln, 145
- dpoint2d_h
 - mln, 145
 - mln::make, 336
- dpoint3d
 - mln, 146
- dpoints_bkd_pixter
 - mln::dpoints_bkd_pixter, 645
- dpoints_fwd_pixter
 - mln::dpoints_fwd_pixter, 647
- dpsite
 - mln::point, 937
 - mln::w_window, 1133
- dpsites_bkd_piter
 - mln::dpsites_bkd_piter, 650
- dpsites_fwd_piter
 - mln::dpsites_fwd_piter, 651
- draw_graph
 - mln::debug, 224, 225
- dual_input_max_tree
 - mln::morpho::tree, 372
- dummy_p_edges
 - mln::make, 337
- dummy_p_vertices
 - mln::make, 337, 338
- duplicate
 - mln, 149
 - mln::border, 189
 - mln::extension, 239
- e_ith_nbh_edge
 - mln::util::graph, 1046
 - mln::util::line_graph, 1053
- e_nmax
 - mln::util::graph, 1046
 - mln::util::line_graph, 1054
- edge
 - mln::p_edges, 854
 - mln::topo, 400
 - mln::util::edge, 1036
 - mln::util::graph, 1046
 - mln::util::line_graph, 1054
- edge_fwd_iter
 - mln::util::graph, 1044
 - mln::util::line_graph, 1053
- edge_image
 - mln::edge_image, 654
 - mln::make, 338, 339
- edge_nbh_edge_fwd_iter
 - mln::util::graph, 1044
 - mln::util::line_graph, 1053
- edge_nbh_t

- mln::edge_image, 653
- edge_win_t
 - mln::edge_image, 653
- edge_with
 - mln::util::vertex, 1087
- edges
 - mln::util::graph, 1046
- edges_set_t
 - mln::util::graph, 1044
- edges_t
 - mln::util::graph, 1044
 - mln::util::line_graph, 1053
- eiter
 - mln::util::array, 1022
 - mln::util::set, 1066
- element
 - mln::box, 551
 - mln::graph_window_if_piter, 749
 - mln::graph_window_piter, 753
 - mln::image1d, 762
 - mln::image2d, 767
 - mln::image3d, 776
 - mln::p_array, 841
 - mln::p_centered, 846
 - mln::p_complex, 849
 - mln::p_edges, 854
 - mln::p_faces, 859
 - mln::p_if, 864
 - mln::p_image, 867
 - mln::p_key, 875
 - mln::p_line2d, 880
 - mln::p_mutable_array_of, 884
 - mln::p_priority, 891
 - mln::p_queue, 896
 - mln::p_queue_fast, 901
 - mln::p_run, 906
 - mln::p_set, 911
 - mln::p_transformed, 915
 - mln::p_vaccess, 920
 - mln::p_vertices, 924
 - mln::util::array, 1022
 - mln::util::set, 1066
 - mln::util::soft_heap, 1072
- elt
 - mln::util::tree_node, 1082
- empty
 - mln::p_queue_fast, 902
- enc
 - mln::value::float01, 1092
 - mln::value::label, 1109
 - mln::value::proxy, 1115
 - mln::value::sign, 1122
- end
 - mln::p_line2d, 881
 - mln::p_run, 907
- enlarge
 - mln::box, 553
- equalize
 - mln::border, 189
 - mln::histo, 266
- equiv
 - mln::value, 424
 - mln::value::float01, 1092
 - mln::value::proxy, 1115
 - mln::value::sign, 1122
- erosion
 - mln::morpho, 356
- erosion_tolerant
 - mln::morpho, 357
- exists_key
 - mln::p_key, 876
- exists_priority
 - mln::p_priority, 892
- extend
 - mln, 150
- extended
 - mln::extended, 655
- extension
 - mln::extension_fun, 658
 - mln::extension_ima, 660
 - mln::extension_val, 662
- extension_fun
 - mln::extension_fun, 657
- extension_ima
 - mln::extension_ima, 660
- extension_val
 - mln::extension_val, 662
- f_hsi_to_rgb_3x8
 - mln::fun::v2v, 245
- f_hsl_to_rgb_3x8
 - mln::fun::v2v, 245
- f_rgb_to_hsi_f
 - mln::fun::v2v, 245
- f_rgb_to_hsl_f
 - mln::fun::v2v, 245
- face
 - mln::complex_psite, 587
 - mln::topo::face, 991
- face_bkd_iter
 - mln::topo::face_bkd_iter, 994
- face_fwd_iter
 - mln::topo::face_fwd_iter, 995
- face_id
 - mln::complex_psite, 587
 - mln::topo::algebraic_face, 976
 - mln::topo::algebraic_n_face, 980
 - mln::topo::face, 992

- mln::topo::n_face, 1003
- faces
 - mln::topo::n_faces_set, 1008
- faces_type
 - mln::topo::n_faces_set, 1008
- fast_median
 - mln::data, 206
- fibonacci_heap
 - mln::util::fibonacci_heap, 1040
- filename
 - mln::debug, 225
- fill
 - mln::border, 189
 - mln::data, 206
 - mln::extension, 239
 - mln::util::array, 1024
- fill_holes
 - mln::labeling, 301
- fill_with_image
 - mln::data, 207
 - mln::data::impl::generic, 219
- fill_with_value
 - mln::data, 207
 - mln::data::impl::generic, 219
- filter
 - mln::morpho::tree::filter, 376
- find
 - mln::border, 190
- first
 - mln::accu::pair, 497
 - mln::accu::stat::min_max, 525
 - mln::util::couple, 1033
 - mln::util::ord_pair, 1061
 - mln::util::site_pair, 1070
- first_accu
 - mln::accu::pair, 497
 - mln::accu::stat::min_max, 525
- first_element
 - mln::util::set, 1066
- flat_image
 - mln::flat_image, 665
- flat_zones
 - mln::labeling, 301
- float01
 - mln::value::float01, 1092
- float01_16
 - mln::value, 422
- float01_8
 - mln::value, 422
- float01_f
 - mln::value::float01_f, 1094
- float_2complex_image3df
 - mln, 146
- flooding
 - mln::morpho::watershed, 379
- foreground
 - mln::labeling, 302
- format
 - mln::debug, 225, 226
- from_to
 - mln::convert, 198
- front
 - mln::p_priority, 892
 - mln::p_queue, 897
 - mln::p_queue_fast, 902
 - mln::util::fibonacci_heap, 1040
- fun
 - mln::p2p_image, 839
- fun_image
 - mln::fun_image, 709
- fun_t
 - mln::p_edges, 854
 - mln::p_vertices, 924
- Function
 - mln::Function, 710
- function
 - mln::p_edges, 856
 - mln::p_transformed, 916
 - mln::p_vertices, 926
- Functions, 99
- fwd_citer
 - mln::topo::complex, 987
- fwd_eiter
 - mln::util::array, 1022
 - mln::util::set, 1066
- fwd_niter
 - mln::doc::Neighborhood, 619
 - mln::graph_elt_mixed_neighborhood, 729
 - mln::graph_elt_neighborhood, 735
 - mln::graph_elt_neighborhood_if, 736
 - mln::neighb, 836
- fwd_piter
 - mln::box, 551
 - mln::doc::Box, 598
 - mln::doc::Fastest_Image, 604
 - mln::doc::Image, 613
 - mln::doc::Site_Set, 629
 - mln::hexa, 755
 - mln::image2d_h, 770
 - mln::p_array, 841
 - mln::p_centered, 846
 - mln::p_complex, 850
 - mln::p_edges, 854
 - mln::p_faces, 859
 - mln::p_if, 864
 - mln::p_image, 867
 - mln::p_key, 875
 - mln::p_line2d, 880

- mln::p_mutable_array_of, 884
- mln::p_priority, 891
- mln::p_queue, 896
- mln::p_queue_fast, 901
- mln::p_run, 906
- mln::p_set, 911
- mln::p_transformed, 915
- mln::p_vaccess, 920
- mln::p_vertices, 924
- fwd_pixter1d
 - mln::fwd_pixter1d, 715
- fwd_pixter2d
 - mln::fwd_pixter2d, 716
- fwd_pixter3d
 - mln::fwd_pixter3d, 718
- fwd_qiter
 - mln::doc::Weighted_Window, 635
 - mln::doc::Window, 637
 - mln::graph_elt_mixed_window, 731
 - mln::graph_elt_window, 739
 - mln::graph_elt_window_if, 743
 - mln::w_window, 1133
 - mln::window, 1151
- fwd_viter
 - mln::doc::Value_Set, 633
 - mln::value::lut_vec, 1112
- gaussian
 - mln::linear, 312, 313
- gaussian_1st_derivative
 - mln::linear, 313
- gaussian_2nd_derivative
 - mln::linear, 313, 314
- gaussian_subsampling
 - mln::subsampling, 392
- general
 - mln::morpho, 357
- geom
 - mln::complex_image, 580
 - mln::p_complex, 851
- get
 - mln::border, 190
 - mln::set, 390
- get_header
 - mln::io::dicom, 270
 - mln::io::dump, 271
 - mln::io::raw, 291
- get_rot
 - mln::registration, 386
- gl16
 - mln::value, 422
- gl8
 - mln::value, 422
- glf
 - mln::value, 422
- gradient
 - mln::morpho, 357
- gradient_external
 - mln::morpho, 357
- gradient_internal
 - mln::morpho, 357
- graph
 - mln::p_edges, 856
 - mln::p_graph_piter, 862
 - mln::p_vertices, 926
 - mln::util::edge, 1036
 - mln::util::graph, 1045
 - mln::util::line_graph, 1054
 - mln::util::vertex, 1087
- Graph based, 109
- graph_element
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_window_piter, 751
 - mln::p_edges, 854
 - mln::p_vertices, 924
- graph_elt_neighborhood_if
 - mln::graph_elt_neighborhood_if, 737
- graph_elt_window_if
 - mln::graph_elt_window_if, 745
- graph_t
 - mln::edge_image, 653
 - mln::p_edges, 855
 - mln::p_vertices, 925
 - mln::util::edge, 1036
 - mln::util::vertex, 1086
 - mln::vertex_image, 1129
- graph_window_if_piter
 - mln::graph_window_if_piter, 749
- graph_window_piter
 - mln::graph_window_piter, 752
- Graphes, 94
- graylevel
 - mln::value::graylevel, 1096
- graylevel_f
 - mln::value::graylevel_f, 1099
- green
 - mln::literal, 320
- grid
 - mln::dpoint, 641
 - mln::point, 937
- h_mat
 - mln::algebra::h_mat, 541
 - mln::make, 340
- h_vec
 - mln::algebra::h_vec, 543
 - mln::point, 937

- has
 - mln::box, 553
 - mln::doc::Box, 598
 - mln::doc::Fastest_Image, 606, 607
 - mln::doc::Image, 615
 - mln::doc::Site_Set, 629
 - mln::doc::Value_Set, 633
 - mln::extension_fun, 658
 - mln::extension_ima, 660
 - mln::extension_val, 663
 - mln::flat_image, 665
 - mln::hexa, 756
 - mln::image1d, 763
 - mln::image2d, 768
 - mln::image2d_h, 771
 - mln::image3d, 776
 - mln::interpolated, 779
 - mln::lazy_image, 792
 - mln::p_array, 843
 - mln::p_centered, 847
 - mln::p_complex, 851
 - mln::p_edges, 856, 857
 - mln::p_if, 865
 - mln::p_image, 869
 - mln::p_key, 877
 - mln::p_line2d, 881
 - mln::p_mutable_array_of, 885
 - mln::p_priority, 892
 - mln::p_queue, 897
 - mln::p_queue_fast, 902
 - mln::p_run, 908
 - mln::p_set, 912
 - mln::p_transformed, 916
 - mln::p_vaccess, 921
 - mln::p_vertices, 927
 - mln::set, 391
 - mln::tr_image, 1014
 - mln::util::line_graph, 1054
 - mln::util::set, 1067
 - mln::value::lut_vec, 1113
 - mln::window, 1152
- has_e
 - mln::util::graph, 1047
 - mln::util::line_graph, 1054
- has_index
 - mln::p_run, 908
- has_v
 - mln::util::graph, 1047
 - mln::util::line_graph, 1055
- height
 - mln::morpho::attribute::sharpness, 830
 - mln::win::cuboid3d, 1141
 - mln::win::rectangle2d, 1148
- hexa
 - mln::hexa, 756
- higher_dim_adj_faces
 - mln::topo::algebraic_face, 976
 - mln::topo::algebraic_n_face, 981
 - mln::topo::face, 992
 - mln::topo::n_face, 1003
- highest_priority
 - mln::p_priority, 892
- histo3d_rgb
 - mln::accu::stat::histo3d_rgb, 510
- hit_or_miss
 - mln::morpho, 358
- hit_or_miss_background_closing
 - mln::morpho, 358
- hit_or_miss_background_opening
 - mln::morpho, 358
- hit_or_miss_closing
 - mln::morpho, 358
- hit_or_miss_opening
 - mln::morpho, 359
- hline2d
 - modwin2d, 114
- hough
 - mln::transform, 408
- i_element
 - mln::p_array, 842
 - mln::p_image, 867
 - mln::p_key, 875
 - mln::p_mutable_array_of, 884
 - mln::p_priority, 891
 - mln::p_queue, 896
 - mln::p_queue_fast, 901
 - mln::p_set, 911
 - mln::p_vaccess, 920
- icp
 - mln::registration, 386, 387
- id
 - mln::graph_window_if_piter, 749
 - mln::graph_window_piter, 753
 - mln::p_graph_piter, 862
 - mln::util::edge, 1036
 - mln::util::vertex, 1087
- id_t
 - mln::util::edge, 1036
 - mln::util::vertex, 1086
- id_value_t
 - mln::util::edge, 1036
 - mln::util::vertex, 1086
- identity
 - mln::literal, 321
- Identity morphers, 97
- ima
 - mln::doc::Generalized_Pixel, 611

- mln::doc::Pixel_Iterator, 622
- mln::fun::x2x::linear, 701
- mln::util::pix, 1063
- image
 - mln::bkd_pixter1d, 545
 - mln::bkd_pixter2d, 546
 - mln::bkd_pixter3d, 547
 - mln::doc::Generalized_Pixel, 610
 - mln::doc::Pixel_Iterator, 622
 - mln::fwd_pixter1d, 715
 - mln::fwd_pixter2d, 716
 - mln::fwd_pixter3d, 717
 - mln::make, 340, 341
 - mln::pw::image, 943
- Image morphers, 96
- image1d
 - mln::image1d, 761
- image2d
 - mln::image2d, 766
 - mln::make, 341
- image2d_h
 - mln::image2d_h, 771
- image3d
 - mln::image3d, 775
 - mln::make, 341
- Images, 95
- implies
 - mln, 150
- inc_face_id
 - mln::topo::algebraic_face, 976
 - mln::topo::algebraic_n_face, 981
 - mln::topo::face, 992
 - mln::topo::n_face, 1003
- inc_n
 - mln::topo::algebraic_face, 976
 - mln::topo::face, 992
- index
 - mln::p_indexed_bkd_piter, 871
 - mln::p_indexed_fwd_piter, 872
- index_of
 - mln::doc::Value_Set, 633
 - mln::value::lut_vec, 1113
- influence_zone_adjacency_graph
 - mln::make, 341
- influence_zone_front
 - mln::transform, 409
- influence_zone_geodesic
 - mln::transform, 409
- influence_zone_geodesic_saturated
 - mln::transform, 410
- init
 - mln::accu::center, 434
 - mln::accu::convolve, 435
 - mln::accu::count_adjacent_vertices, 437
 - mln::accu::count_labels, 439
 - mln::accu::count_value, 440
 - mln::accu::label_used, 443
 - mln::accu::logic::land, 445
 - mln::accu::logic::land_basic, 446
 - mln::accu::logic::lor, 448
 - mln::accu::logic::lor_basic, 449
 - mln::accu::maj_h, 451
 - mln::accu::math::count, 452
 - mln::accu::math::inf, 454
 - mln::accu::math::sum, 455
 - mln::accu::math::sup, 456
 - mln::accu::max_site, 458
 - mln::accu::nil, 493
 - mln::accu::p, 494
 - mln::accu::pair, 497
 - mln::accu::rms, 499
 - mln::accu::shape::bbox, 500
 - mln::accu::shape::height, 502
 - mln::accu::shape::volume, 504
 - mln::accu::stat::deviation, 508
 - mln::accu::stat::histo3d_rgb, 510
 - mln::accu::stat::max, 512
 - mln::accu::stat::max_h, 513
 - mln::accu::stat::mean, 515
 - mln::accu::stat::median_h, 519
 - mln::accu::stat::min, 521
 - mln::accu::stat::min_h, 523
 - mln::accu::stat::min_max, 525
 - mln::accu::stat::rank, 527
 - mln::accu::stat::rank < bool >, 528
 - mln::accu::stat::rank_high_quant, 530
 - mln::accu::stat::var, 532
 - mln::accu::stat::variance, 534
 - mln::accu::tuple, 536
 - mln::accu::val, 538
 - mln::doc::Accumulator, 596
 - mln::morpho::attribute::card, 825
 - mln::morpho::attribute::count_adjacent_vertices, 827
 - mln::morpho::attribute::height, 828
 - mln::morpho::attribute::sharpness, 830
 - mln::morpho::attribute::sum, 832
 - mln::morpho::attribute::volume, 834
 - mln::p_run, 908
- initialize
 - mln, 150
- insert
 - mln::p_array, 843
 - mln::p_image, 869
 - mln::p_key, 877
 - mln::p_mutable_array_of, 885
 - mln::p_priority, 892
 - mln::p_queue, 897

- mln::p_queue_fast, 903
- mln::p_set, 912
- mln::p_vaccess, 921
- mln::util::set, 1067
- mln::w_window, 1134
- mln::window, 1152
- int_s
 - mln::value::int_s, 1102
- int_s16
 - mln::value, 422
- int_s32
 - mln::value, 423
- int_s8
 - mln::value, 423
- int_u
 - mln::value::int_u, 1104
- int_u12
 - mln::value, 423
- int_u16
 - mln::value, 423
- int_u32
 - mln::value, 423
- int_u8
 - mln::value, 423
- int_u8_1complex_image2d
 - mln, 146
- int_u8_2complex_image2d
 - mln, 146
- int_u8_2complex_image3df
 - mln, 146
- int_u_sat
 - mln::value::int_u_sat, 1106
- inter
 - mln::Box, 560
 - mln::Site_Set, 952
- interpolated
 - mln::interpolated, 779
- inv
 - mln::fun::x2x::rotation, 704
 - mln::fun::x2x::translation, 707
- invalidate
 - mln::complex_psite, 587
 - mln::doc::Iterator, 618
 - mln::doc::Pixel_Iterator, 623
 - mln::doc::Site_Iterator, 627
 - mln::doc::Value_Iterator, 631
 - mln::dpoints_bkd_pixter, 645
 - mln::dpoints_fwd_pixter, 648
 - mln::p_edges, 857
 - mln::p_vertices, 927
 - mln::topo::algebraic_face, 976
 - mln::topo::algebraic_n_face, 981
 - mln::topo::face, 992
 - mln::topo::n_face, 1004
 - mln::util::branch_iter, 1029
 - mln::util::branch_iter_ind, 1030
 - mln::util::edge, 1037
 - mln::util::vertex, 1087
- invert
 - mln::fun::x2x::rotation, 703
 - mln::fun::x2x::translation, 706
- iota
 - mln::debug, 226
- is_centered
 - mln::doc::Weighted_Window, 636
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_elt_window_if, 745
 - mln::graph_window_base, 747
 - mln::window, 1153
- is_empty
 - mln::Box, 559
 - mln::box, 554
 - mln::doc::Weighted_Window, 636
 - mln::graph_elt_mixed_window, 733
 - mln::graph_elt_window, 741
 - mln::graph_elt_window_if, 745
 - mln::graph_window_base, 747
 - mln::util::array, 1024
 - mln::util::fibonacci_heap, 1040
 - mln::util::set, 1067
 - mln::util::soft_heap, 1073
 - mln::window, 1153
- is_facet
 - mln::topo, 400
- is_subgraph_of
 - mln::util::graph, 1047
 - mln::util::line_graph, 1055
- is_symmetric
 - mln::graph_elt_mixed_window, 733
 - mln::graph_elt_window, 741
 - mln::graph_elt_window_if, 745
 - mln::graph_window_base, 748
 - mln::w_window, 1134
 - mln::window, 1153
- is_valid
 - mln::accu::center, 434
 - mln::accu::convolve, 435
 - mln::accu::count_adjacent_vertices, 437
 - mln::accu::count_labels, 439
 - mln::accu::count_value, 440
 - mln::accu::histo, 442
 - mln::accu::label_used, 443
 - mln::accu::logic::land, 445
 - mln::accu::logic::land_basic, 446
 - mln::accu::logic::lor, 448
 - mln::accu::logic::lor_basic, 449
 - mln::accu::maj_h, 451

- mln::accu::math::count, 452
- mln::accu::math::inf, 454
- mln::accu::math::sum, 455
- mln::accu::math::sup, 456
- mln::accu::max_site, 458
- mln::accu::nil, 493
- mln::accu::p, 494
- mln::accu::pair, 497
- mln::accu::rms, 499
- mln::accu::shape::bbox, 500
- mln::accu::shape::height, 502
- mln::accu::shape::volume, 504
- mln::accu::stat::deviation, 508
- mln::accu::stat::histo3d_rgb, 510
- mln::accu::stat::max, 512
- mln::accu::stat::max_h, 513
- mln::accu::stat::mean, 515
- mln::accu::stat::median_alt, 517
- mln::accu::stat::median_h, 519
- mln::accu::stat::min, 521
- mln::accu::stat::min_h, 523
- mln::accu::stat::min_max, 525
- mln::accu::stat::rank, 527
- mln::accu::stat::rank < bool >, 528
- mln::accu::stat::rank_high_quant, 530
- mln::accu::stat::var, 532
- mln::accu::stat::variance, 534
- mln::accu::tuple, 536
- mln::accu::val, 538
- mln::box, 554
- mln::complex_psite, 587
- mln::doc::Fastest_Image, 607
- mln::doc::Image, 616
- mln::doc::Iterator, 618
- mln::doc::Pixel_Iterator, 623
- mln::doc::Site_Iterator, 627
- mln::doc::Value_Iterator, 631
- mln::dpoints_bkd_pixter, 645
- mln::dpoints_fwd_pixter, 648
- mln::graph_elt_mixed_window, 733
- mln::graph_elt_window, 741
- mln::graph_elt_window_if, 746
- mln::graph_window_base, 748
- mln::interpolated, 779
- mln::morpho::attribute::card, 825
- mln::morpho::attribute::count_adjacent_vertices, 827
- mln::morpho::attribute::height, 828
- mln::morpho::attribute::sharpness, 830
- mln::morpho::attribute::sum, 832
- mln::morpho::attribute::volume, 834
- mln::p_array, 843
- mln::p_centered, 847
- mln::p_complex, 851
- mln::p_edges, 857
- mln::p_faces, 860
- mln::p_if, 865
- mln::p_image, 869
- mln::p_key, 877
- mln::p_line2d, 882
- mln::p_mutable_array_of, 885
- mln::p_priority, 893
- mln::p_queue, 898
- mln::p_queue_fast, 903
- mln::p_run, 908
- mln::p_set, 912
- mln::p_transformed, 916
- mln::p_vaccess, 921
- mln::p_vertices, 927
- mln::pixel, 929
- mln::topo::algebraic_face, 976
- mln::topo::algebraic_n_face, 981
- mln::topo::face, 992
- mln::topo::n_face, 1004
- mln::tr_image, 1014
- mln::util::branch_iter, 1029
- mln::util::branch_iter_ind, 1031
- mln::util::edge, 1037
- mln::util::fibonacci_heap, 1041
- mln::util::soft_heap, 1073
- mln::util::vertex, 1087
- mln::value::stack_image, 1125
- iter
 - mln::complex_neighborhood_bkd_piter, 583
 - mln::complex_neighborhood_fwd_piter, 585
 - mln::complex_window_bkd_piter, 589
 - mln::complex_window_fwd_piter, 591
- iter_type
 - mln::complex_neighborhood_bkd_piter, 582
 - mln::complex_neighborhood_fwd_piter, 584
 - mln::complex_window_bkd_piter, 589
 - mln::complex_window_fwd_piter, 590
- ith_nbh_edge
 - mln::util::edge, 1037
 - mln::util::vertex, 1087
- ith_nbh_vertex
 - mln::util::vertex, 1088
- k
 - mln::accu::stat::rank, 527
- key
 - mln::p_key, 877
- keys
 - mln::p_key, 877
- ll
 - mln::norm, 382
- ll_distance

- mln::norm, [382](#)
- l2
 - mln::norm, [382](#)
- l2_distance
 - mln::norm, [382](#)
- label
 - mln::value::label, [1110](#)
- label_16
 - mln::value, [423](#)
- label_32
 - mln::value, [423](#)
- label_8
 - mln::value, [423](#)
- labeled_image
 - mln::labeled_image, [786](#)
- labeled_image_base
 - mln::labeled_image_base, [789](#)
- labeling
 - mln::graph, [264](#)
- laplacian
 - mln::morpho, [359](#)
- larger_than
 - mln, [151](#)
- last
 - mln::util::array, [1024](#)
- last_coord
 - mln::point, [939](#)
- last_element
 - mln::util::set, [1068](#)
- lazy_image
 - mln::lazy_image, [792](#)
- ldlt_decomp
 - mln::algebra, [173](#)
- ldlt_solve
 - mln::algebra, [173](#)
- lemmings
 - mln::util, [414](#)
- len
 - mln::Box, [559](#)
 - mln::box, [554](#)
- length
 - mln::p_run, [908](#)
 - mln::win::backdiag2d, [1138](#)
 - mln::win::cube3d, [1140](#)
 - mln::win::diag2d, [1143](#)
 - mln::win::line, [1144](#)
 - mln::win::octagon2d, [1146](#)
- light_gray
 - mln::literal, [321](#)
- lime
 - mln::literal, [321](#)
- line
 - mln::accu, [165](#)
 - mln::draw, [234](#)
 - mln::win::line, [1144](#)
- line_gradient
 - mln::morpho, [359](#)
- linear
 - mln::fun::x2x::linear, [701](#)
- linfty
 - mln::norm, [382](#)
- linfty_distance
 - mln::norm, [382](#)
- load
 - mln::io::cloud, [269](#)
 - mln::io::dicom, [270](#)
 - mln::io::dump, [271](#)
 - mln::io::fits, [272](#), [273](#)
 - mln::io::fld, [274](#)
 - mln::io::magick, [275](#)
 - mln::io::off, [276](#)
 - mln::io::pbm, [278](#)
 - mln::io::pbms, [279](#)
 - mln::io::pfm, [281](#)
 - mln::io::pgm, [282](#)
 - mln::io::pgms, [283](#)
 - mln::io::plot, [284](#)
 - mln::io::pnm, [286](#)
 - mln::io::pnms, [287](#), [288](#)
 - mln::io::ppm, [288](#), [289](#)
 - mln::io::ppms, [290](#)
 - mln::io::raw, [291](#)
 - mln::io::tiff, [292](#)
- load_raw_2d
 - mln::io::pnm, [286](#)
- lower_dim_adj_faces
 - mln::topo::algebraic_face, [977](#)
 - mln::topo::algebraic_n_face, [981](#)
 - mln::topo::face, [992](#)
 - mln::topo::n_face, [1004](#)
- lowest_priority
 - mln::p_priority, [893](#)
- lut_vec
 - mln::value::lut_vec, [1113](#)
- lvalue
 - mln::complex_image, [580](#)
 - mln::decorated_image, [592](#)
 - mln::doc::Fastest_Image, [604](#)
 - mln::doc::Image, [614](#)
 - mln::doc::Pixel_Iterator, [622](#)
 - mln::flat_image, [664](#)
 - mln::fun_image, [708](#)
 - mln::hexa, [755](#)
 - mln::image1d, [761](#)
 - mln::image2d, [766](#)
 - mln::image2d_h, [770](#)
 - mln::image3d, [774](#)
 - mln::interpolated, [778](#)

- [mln::lazy_image, 792](#)
 - [mln::tr_image, 1013](#)
 - [mln::value::stack_image, 1124](#)
 - [mln::violent_cast_image, 1131](#)
- [magenta](#)
 - [mln::literal, 321](#)
- [main_branch](#)
 - [mln::util::tree, 1079](#)
- [make_algebraic_face](#)
 - [mln::topo, 400](#)
- [make_algebraic_n_face](#)
 - [mln::topo, 400](#)
- [make_debug_graph_image](#)
 - [mln, 151](#)
- [make_greater_point](#)
 - [mln::util, 415](#)
- [make_greater_psite](#)
 - [mln::util, 415](#)
- [mask](#)
 - [mln::graph_elt_neighborhood_if, 737](#)
 - [mln::graph_elt_window_if, 746](#)
- [mask_t](#)
 - [mln::graph_elt_window_if, 744](#)
- [mat](#)
 - [mln::make, 342](#)
- [max](#)
 - [mln::literal, 321](#)
 - [mln::morpho::tree::filter, 377](#)
- [max_col](#)
 - [mln::geom, 254, 255](#)
- [max_component](#)
 - [mln::io::pnm, 286](#)
- [max_ind](#)
 - [mln::geom, 255](#)
- [max_row](#)
 - [mln::geom, 255](#)
- [max_sli](#)
 - [mln::geom, 255](#)
- [max_tree](#)
 - [mln::morpho::tree, 372](#)
- [mean](#)
 - [mln::accu::stat::var, 532](#)
 - [mln::accu::stat::variance, 534](#)
 - [mln::estim, 236](#)
- [mean_t](#)
 - [mln::accu::stat::var, 532](#)
- [median](#)
 - [mln::data, 208](#)
 - [mln::data::approx, 214, 215](#)
 - [mln::data::naive, 222](#)
- [medium_gray](#)
 - [mln::literal, 321](#)
- [memory_size](#)
 - [mln::box, 554](#)
 - [mln::p_array, 843](#)
 - [mln::p_centered, 848](#)
 - [mln::p_edges, 857](#)
 - [mln::p_if, 865](#)
 - [mln::p_image, 869](#)
 - [mln::p_key, 877](#)
 - [mln::p_line2d, 882](#)
 - [mln::p_mutable_array_of, 885](#)
 - [mln::p_priority, 893](#)
 - [mln::p_queue, 898](#)
 - [mln::p_queue_fast, 903](#)
 - [mln::p_run, 908](#)
 - [mln::p_set, 912](#)
 - [mln::p_transformed, 916](#)
 - [mln::p_vaccess, 921](#)
 - [mln::p_vertices, 927](#)
 - [mln::util::array, 1024](#)
 - [mln::util::set, 1068](#)
- [merge](#)
 - [mln::box, 554](#)
- [mesh](#)
 - [mln::doc::Point_Site, 624](#)
- [mesh_corner_point_area](#)
 - [mln::geom, 255](#)
- [mesh_curvature](#)
 - [mln::geom, 256](#)
- [mesh_normal](#)
 - [mln::geom, 256](#)
- [meyer_wst](#)
 - [mln::morpho, 359, 360](#)
- [min](#)
 - [mln::arith, 177](#)
 - [mln::literal, 321](#)
 - [mln::morpho, 360](#)
 - [mln::morpho::tree::filter, 377](#)
- [min_col](#)
 - [mln::geom, 256, 257](#)
- [min_ind](#)
 - [mln::geom, 257](#)
- [min_inplace](#)
 - [mln::arith, 178](#)
 - [mln::morpho, 360](#)
- [min_max](#)
 - [mln::estim, 236](#)
- [min_row](#)
 - [mln::geom, 257](#)
- [min_sli](#)
 - [mln::geom, 257](#)
- [min_tree](#)
 - [mln::morpho::tree, 372](#)
- [minus](#)
 - [mln::arith, 178, 179](#)
 - [mln::morpho, 360](#)

- minus_cst
 - mln::arith, 179, 180
- minus_cst_inplace
 - mln::arith, 180
- minus_infty
 - mln::point, 939
- minus_inplace
 - mln::arith, 180
- mirror
 - mln::border, 190
- mln, 121
 - a_point_of, 148
 - apply_p2p, 149
 - before, 162
 - bin_1complex_image2d, 144
 - bin_2complex_image3df, 144
 - box1d, 144
 - box2d, 144
 - box2d_h, 145
 - box3d, 145
 - compose, 149
 - discrete_plane_1complex_geometry, 145
 - discrete_plane_2complex_geometry, 145
 - dpoint1d, 145
 - dpoint2d, 145
 - dpoint2d_h, 145
 - dpoint3d, 146
 - duplicate, 149
 - extend, 150
 - float_2complex_image3df, 146
 - implies, 150
 - initialize, 150
 - int_u8_1complex_image2d, 146
 - int_u8_2complex_image2d, 146
 - int_u8_2complex_image3df, 146
 - larger_than, 151
 - make_debug_graph_image, 151
 - mln_exact, 151
 - mln_gen_complex_neighborhood, 151, 152
 - mln_gen_complex_window, 152, 153
 - mln_gen_complex_window_p, 153
 - mln_regular, 153
 - mln_trait_op_geq, 154
 - mln_trait_op_greater, 154
 - mln_trait_op_leq, 154
 - mln_trait_op_neq, 154
 - operator<, 156, 157
 - operator<<, 157, 158
 - operator<=, 158, 159
 - operator*, 155
 - operator++, 155
 - operator-, 156
 - operator--, 156
 - operator==, 159, 160
 - p_run2d, 146
 - p_runs2d, 146
 - point1d, 147
 - point1df, 147
 - point2d, 147
 - point2d_h, 147
 - point2df, 147
 - point3d, 147
 - point3df, 147
 - primary, 161
 - ptransform, 162
 - rgb8_2complex_image3df, 147
 - sagittal_dec, 162
 - space_2complex_geometry, 148
 - unsigned_2complex_image3df, 148
 - up, 162
 - vec2d_d, 148
 - vec2d_f, 148
 - vec3d_d, 148
 - vec3d_f, 148
- mln::accu, 162
 - compute, 164
 - line, 165
 - mln_meta_accu_result, 165
 - take, 165
- mln::accu::center, 433
 - init, 434
 - is_valid, 434
 - nsites, 434
 - take_as_init, 434
 - take_n_times, 434
 - to_result, 434
- mln::accu::convolve, 435
 - init, 435
 - is_valid, 435
 - take_as_init, 436
 - take_n_times, 436
 - to_result, 436
- mln::accu::count_adjacent_vertices, 436
 - init, 437
 - is_valid, 437
 - set_value, 437
 - take_as_init, 437
 - take_n_times, 437
 - to_result, 438
- mln::accu::count_labels, 438
 - init, 439
 - is_valid, 439
 - set_value, 439
 - take_as_init, 439
 - take_n_times, 439
 - to_result, 439
- mln::accu::count_value, 439
 - init, 440

- [is_valid, 440](#)
 - [set_value, 440](#)
 - [take_as_init, 440](#)
 - [take_n_times, 441](#)
 - [to_result, 441](#)
- [mln::accu::histo, 441](#)
 - [is_valid, 442](#)
 - [take, 442](#)
 - [take_as_init, 442](#)
 - [take_n_times, 442](#)
 - [vect, 442](#)
- [mln::accu::image, 166](#)
- [mln::accu::impl, 166](#)
- [mln::accu::label_used, 442](#)
 - [init, 443](#)
 - [is_valid, 443](#)
 - [take, 443](#)
 - [take_as_init, 443](#)
 - [take_n_times, 444](#)
 - [to_result, 444](#)
- [mln::accu::logic, 166](#)
- [mln::accu::logic::land, 444](#)
 - [init, 445](#)
 - [is_valid, 445](#)
 - [take_as_init, 445](#)
 - [take_n_times, 445](#)
 - [to_result, 445](#)
- [mln::accu::logic::land_basic, 445](#)
 - [can_stop, 446](#)
 - [init, 446](#)
 - [is_valid, 446](#)
 - [take_as_init, 446](#)
 - [take_n_times, 447](#)
 - [to_result, 447](#)
- [mln::accu::logic::lor, 447](#)
 - [init, 448](#)
 - [is_valid, 448](#)
 - [take_as_init, 448](#)
 - [take_n_times, 448](#)
 - [to_result, 448](#)
- [mln::accu::logic::lor_basic, 448](#)
 - [can_stop, 449](#)
 - [init, 449](#)
 - [is_valid, 449](#)
 - [take_as_init, 449](#)
 - [take_n_times, 449](#)
 - [to_result, 450](#)
- [mln::accu::maj_h, 450](#)
 - [init, 451](#)
 - [is_valid, 451](#)
 - [take_as_init, 451](#)
 - [take_n_times, 451](#)
 - [to_result, 451](#)
- [mln::accu::math, 167](#)
 - [mln::accu::math::count, 451](#)
 - [init, 452](#)
 - [is_valid, 452](#)
 - [set_value, 452](#)
 - [take_as_init, 452](#)
 - [take_n_times, 452](#)
 - [to_result, 453](#)
 - [mln::accu::math::inf, 453](#)
 - [init, 454](#)
 - [is_valid, 454](#)
 - [take_as_init, 454](#)
 - [take_n_times, 454](#)
 - [to_result, 454](#)
 - [mln::accu::math::sum, 454](#)
 - [init, 455](#)
 - [is_valid, 455](#)
 - [take_as_init, 455](#)
 - [take_n_times, 455](#)
 - [to_result, 455](#)
 - [mln::accu::math::sup, 456](#)
 - [init, 456](#)
 - [is_valid, 456](#)
 - [take_as_init, 457](#)
 - [take_n_times, 457](#)
 - [to_result, 457](#)
 - [mln::accu::max_site, 457](#)
 - [init, 458](#)
 - [is_valid, 458](#)
 - [take_as_init, 458](#)
 - [take_n_times, 458](#)
 - [to_result, 458](#)
 - [mln::accu::meta::center, 459](#)
 - [mln::accu::meta::count_adjacent_vertices, 459](#)
 - [mln::accu::meta::count_labels, 460](#)
 - [mln::accu::meta::count_value, 461](#)
 - [mln::accu::meta::histo, 462](#)
 - [mln::accu::meta::label_used, 463](#)
 - [mln::accu::meta::logic, 167](#)
 - [mln::accu::meta::logic::land, 464](#)
 - [mln::accu::meta::logic::land_basic, 465](#)
 - [mln::accu::meta::logic::lor, 466](#)
 - [mln::accu::meta::logic::lor_basic, 467](#)
 - [mln::accu::meta::maj_h, 468](#)
 - [mln::accu::meta::math, 168](#)
 - [mln::accu::meta::math::count, 469](#)
 - [mln::accu::meta::math::inf, 470](#)
 - [mln::accu::meta::math::sum, 471](#)
 - [mln::accu::meta::math::sup, 472](#)
 - [mln::accu::meta::max_site, 473](#)
 - [mln::accu::meta::nil, 474](#)
 - [mln::accu::meta::p, 475](#)
 - [mln::accu::meta::pair, 476](#)
 - [mln::accu::meta::rms, 477](#)
 - [mln::accu::meta::shape, 168](#)

- mln::accu::meta::shape::bbox, 478
- mln::accu::meta::shape::height, 479
- mln::accu::meta::shape::volume, 480
- mln::accu::meta::stat, 169
- mln::accu::meta::stat::max, 481
- mln::accu::meta::stat::max_h, 482
- mln::accu::meta::stat::mean, 483
- mln::accu::meta::stat::median_alt, 484
- mln::accu::meta::stat::median_h, 485
- mln::accu::meta::stat::min, 486
- mln::accu::meta::stat::min_h, 487
- mln::accu::meta::stat::rank, 488
- mln::accu::meta::stat::rank_high_quant, 489
- mln::accu::meta::tuple, 490
- mln::accu::meta::val, 491
- mln::accu::nil, 492
 - init, 493
 - is_valid, 493
 - take_as_init, 493
 - take_n_times, 493
 - to_result, 493
- mln::accu::p, 494
 - init, 494
 - is_valid, 494
 - take_as_init, 495
 - take_n_times, 495
 - to_result, 495
- mln::accu::pair, 495
 - first, 497
 - first_accu, 497
 - init, 497
 - is_valid, 497
 - second, 497
 - second_accu, 497
 - take_as_init, 498
 - take_n_times, 498
 - to_result, 498
- mln::accu::rms, 498
 - init, 499
 - is_valid, 499
 - take_as_init, 499
 - take_n_times, 499
 - to_result, 499
- mln::accu::shape, 170
- mln::accu::shape::bbox, 500
 - init, 500
 - is_valid, 500
 - take_as_init, 500
 - take_n_times, 501
 - to_result, 501
- mln::accu::shape::height, 501
 - argument, 502
 - init, 502
 - is_valid, 502
 - set_value, 502
 - take_as_init, 503
 - take_n_times, 503
 - to_result, 503
 - value, 502
- mln::accu::shape::volume, 503
 - argument, 504
 - init, 504
 - is_valid, 504
 - set_value, 505
 - take_as_init, 505
 - take_n_times, 505
 - to_result, 505
 - value, 504
- mln::accu::site_set::rectangularity, 505
 - area, 506
 - bbox, 506
 - rectangularity, 506
 - take_as_init, 506
 - take_n_times, 507
 - to_result, 507
- mln::accu::stat, 170
 - operator==, 171
- mln::accu::stat::deviation, 507
 - init, 508
 - is_valid, 508
 - take_as_init, 508
 - take_n_times, 508
 - to_result, 508
- mln::accu::stat::histo3d_rgb, 509
 - histo3d_rgb, 510
 - init, 510
 - is_valid, 510
 - take, 510
 - take_as_init, 511
 - take_n_times, 511
 - to_result, 511
- mln::accu::stat::max, 511
 - init, 512
 - is_valid, 512
 - set_value, 512
 - take_as_init, 512
 - take_n_times, 512
 - to_result, 512
- mln::accu::stat::max_h, 513
 - init, 513
 - is_valid, 513
 - take_as_init, 514
 - take_n_times, 514
 - to_result, 514
- mln::accu::stat::mean, 514
 - count, 515
 - init, 515
 - is_valid, 515

- sum, 515
- take_as_init, 515
- take_n_times, 516
- to_result, 516
- mln::accu::stat::median_alt, 516
 - is_valid, 517
 - take, 517
 - take_as_init, 517
 - take_n_times, 517
 - to_result, 517
- mln::accu::stat::median_h, 518
 - init, 519
 - is_valid, 519
 - take_as_init, 519
 - take_n_times, 519
 - to_result, 519
- mln::accu::stat::meta::deviation, 519
- mln::accu::stat::min, 520
 - init, 521
 - is_valid, 521
 - set_value, 521
 - take_as_init, 521
 - take_n_times, 521
 - to_result, 522
- mln::accu::stat::min_h, 522
 - init, 523
 - is_valid, 523
 - take_as_init, 523
 - take_n_times, 523
 - to_result, 523
- mln::accu::stat::min_max, 523
 - first, 525
 - first_accu, 525
 - init, 525
 - is_valid, 525
 - second, 525
 - second_accu, 525
 - take_as_init, 525
 - take_n_times, 526
 - to_result, 526
- mln::accu::stat::rank, 526
 - init, 527
 - is_valid, 527
 - k, 527
 - take_as_init, 527
 - take_n_times, 527
 - to_result, 527
- mln::accu::stat::rank< bool >, 528
 - init, 528
 - is_valid, 528
 - take_as_init, 529
 - take_n_times, 529
 - to_result, 529
- mln::accu::stat::rank_high_quant, 529
- init, 530
- is_valid, 530
- take_as_init, 530
- take_n_times, 530
- to_result, 530
- mln::accu::stat::var, 531
 - init, 532
 - is_valid, 532
 - mean, 532
 - mean_t, 532
 - n_items, 532
 - take_as_init, 532
 - take_n_times, 532
 - to_result, 532
 - variance, 533
- mln::accu::stat::variance, 533
 - init, 534
 - is_valid, 534
 - mean, 534
 - n_items, 534
 - standard_deviation, 534
 - sum, 535
 - take_as_init, 535
 - take_n_times, 535
 - to_result, 535
 - var, 535
- mln::accu::tuple, 535
 - init, 536
 - is_valid, 536
 - take_as_init, 536
 - take_n_times, 537
 - to_result, 537
- mln::accu::val, 537
 - init, 538
 - is_valid, 538
 - take_as_init, 538
 - take_n_times, 538
 - to_result, 538
- mln::Accumulator, 538
 - take_as_init, 540
 - take_n_times, 540
- mln::algebra, 172
 - ldlt_decomp, 173
 - ldlt_solve, 173
 - operator*, 173
 - vprod, 173
- mln::algebra::h_mat, 540
 - _1, 542
 - h_mat, 541
 - t, 542
- mln::algebra::h_vec, 542
 - h_vec, 543
 - operator mat< n, 1, U >, 543
 - origin, 544

- t, [543](#)
- to_vec, [544](#)
- zero, [544](#)
- mln::arith, [173](#)
 - diff_abs, [176](#)
 - div, [176](#)
 - div_cst, [177](#)
 - div_inplace, [177](#)
 - min, [177](#)
 - min_inplace, [178](#)
 - minus, [178](#), [179](#)
 - minus_cst, [179](#), [180](#)
 - minus_cst_inplace, [180](#)
 - minus_inplace, [180](#)
 - plus, [181](#), [182](#)
 - plus_cst, [182](#), [183](#)
 - plus_cst_inplace, [183](#)
 - plus_inplace, [184](#)
 - revert, [184](#)
 - revert_inplace, [184](#)
 - times, [185](#)
 - times_cst, [185](#)
 - times_inplace, [185](#)
- mln::arith::impl, [186](#)
- mln::arith::impl::generic, [186](#)
- mln::binarization, [187](#)
 - binarization, [187](#)
 - threshold, [187](#)
- mln::bkd_pixter1d, [544](#)
 - bkd_pixter1d, [545](#)
 - image, [545](#)
 - next, [545](#)
- mln::bkd_pixter2d, [545](#)
 - bkd_pixter2d, [546](#)
 - image, [546](#)
 - next, [546](#)
- mln::bkd_pixter3d, [547](#)
 - bkd_pixter3d, [548](#)
 - image, [547](#)
 - next, [548](#)
- mln::border, [188](#)
 - adjust, [188](#)
 - duplicate, [189](#)
 - equalize, [189](#)
 - fill, [189](#)
 - find, [190](#)
 - get, [190](#)
 - mirror, [190](#)
 - resize, [191](#)
- mln::border::impl, [191](#)
- mln::border::impl::generic, [192](#)
- mln::Box, [556](#)
 - bbox, [559](#)
 - diff, [560](#)
 - inter, [560](#)
 - is_empty, [559](#)
 - len, [559](#)
 - nsites, [559](#)
 - operator<, [560](#)
 - operator<<, [560](#)
 - operator<=, [560](#), [561](#)
 - operator==, [561](#)
 - sym_diff, [561](#)
 - uni, [561](#)
 - unique, [561](#)
- mln::box, [548](#)
 - bbox, [553](#)
 - bkd_piter, [551](#)
 - box, [552](#)
 - crop_wrt, [553](#)
 - element, [551](#)
 - enlarge, [553](#)
 - fwd_piter, [551](#)
 - has, [553](#)
 - is_empty, [554](#)
 - is_valid, [554](#)
 - len, [554](#)
 - memory_size, [554](#)
 - merge, [554](#)
 - nsites, [554](#)
 - operator<<, [556](#)
 - pcenter, [555](#)
 - piter, [552](#)
 - pmax, [555](#)
 - pmin, [555](#)
 - psite, [552](#)
 - site, [552](#)
 - to_larger, [555](#)
- mln::box_runend_piter, [562](#)
 - next, [562](#)
 - run_length, [562](#)
- mln::box_runstart_piter, [563](#)
 - box_runstart_piter, [563](#)
 - next, [563](#)
 - run_length, [564](#)
- mln::Browsing, [564](#)
- mln::canvas, [192](#)
 - distance_front, [193](#)
 - distance_geodesic, [193](#)
- mln::canvas::browsing, [193](#)
- mln::canvas::browsing::backdiagonal2d_t, [565](#)
- mln::canvas::browsing::breadth_first_search_t, [566](#)
- mln::canvas::browsing::depth_first_search_t, [566](#)
- mln::canvas::browsing::diagonal2d_t, [567](#)
- mln::canvas::browsing::dir_struct_elt_incr_-
update_t, [568](#)
- mln::canvas::browsing::directional_t, [569](#)
- mln::canvas::browsing::fwd_t, [571](#)

- mln::canvas::browsing::hyper_directional_t, 572
- mln::canvas::browsing::snake_fwd_t, 574
- mln::canvas::browsing::snake_generic_t, 575
- mln::canvas::browsing::snake_vert_t, 577
- mln::canvas::chamfer, 578
- mln::canvas::impl, 194
- mln::canvas::labeling, 194
 - blobs, 195
- mln::canvas::labeling::impl, 195
- mln::canvas::morpho, 195
- mln::category< R(*) (A) >, 578
- mln::complex_image, 579
 - complex_image, 581
 - dim, 581
 - domain, 581
 - geom, 580
 - lvalue, 580
 - operator(), 581
 - rvalue, 580
 - skeleton, 580
 - value, 580
 - values, 581
- mln::complex_neighborhood_bkd_piter, 582
 - complex_neighborhood_bkd_piter, 583
 - iter, 583
 - iter_type, 582
 - next, 583
 - psite, 582
- mln::complex_neighborhood_fwd_piter, 583
 - complex_neighborhood_fwd_piter, 584
 - iter, 585
 - iter_type, 584
 - next, 585
 - psite, 584
- mln::complex_psite, 585
 - change_target, 586
 - complex_psite, 586
 - face, 587
 - face_id, 587
 - invalidate, 587
 - is_valid, 587
 - n, 587
 - site_set, 587
- mln::complex_window_bkd_piter, 588
 - complex_window_bkd_piter, 589
 - iter, 589
 - iter_type, 589
 - next, 589
 - psite, 589
- mln::complex_window_fwd_piter, 590
 - complex_window_fwd_piter, 591
 - iter, 591
 - iter_type, 590
 - next, 591
 - psite, 590
- mln::convert, 196
 - from_to, 198
 - mln_image_from_grid, 198, 199
 - mln_window, 199
 - to, 199
 - to_dpoint, 199
 - to_fun, 199, 202
 - to_image, 199
 - to_p_array, 200
 - to_p_set, 200, 201
 - to_qimage, 201
 - to_upper_window, 201
 - to_window, 201, 202
- mln::data, 202
 - abs, 204
 - abs_inplace, 204
 - apply, 205
 - compute, 205
 - convert, 206
 - fast_median, 206
 - fill, 206
 - fill_with_image, 207
 - fill_with_value, 207
 - median, 208
 - mln_meta_accu_result, 208
 - paste, 208
 - paste_without_localization, 209
 - replace, 209
 - saturate, 209
 - saturate_inplace, 210
 - sort_offsets_increasing, 210
 - sort_psites_decreasing, 210
 - sort_psites_increasing, 210
 - stretch, 211
 - to_enc, 211
 - transform, 211, 212
 - transform_inplace, 212
 - update, 213
 - wrap, 213
- mln::data::approx, 214
 - median, 214, 215
- mln::data::approx::impl, 215
- mln::data::impl, 215
 - paste_without_localization_fast, 216
 - paste_without_localization_fastest, 216
 - paste_without_localization_lines, 217
 - stretch, 217
 - transform_inplace_lowq, 218
 - update_fastest, 218
- mln::data::impl::generic, 218
 - fill_with_image, 219
 - fill_with_value, 219
 - paste, 220

- transform, 220
- transform_inplace, 221
- update, 221
- mln::data::naive, 221
 - median, 222
- mln::data::naive::impl, 222
- mln::debug, 223
 - draw_graph, 224, 225
 - filename, 225
 - format, 225, 226
 - iota, 226
 - mosaic, 226
 - println, 226, 227
 - println_with_border, 227
 - put_word, 227
 - slices_2d, 227
 - superpose, 227, 228
 - z_order, 228
- mln::debug::impl, 228
- mln::decorated_image, 591
 - decorated_image, 593
 - decoration, 593
 - lvalue, 592
 - operator decorated_image< const I, D >, 593
 - operator(), 593, 594
 - psite, 592
 - rvalue, 593
 - skeleton, 593
- mln::def, 229
 - coord, 229
 - coordf, 229
- mln::Delta_Point_Site, 594
- mln::Delta_Point_Site< void >, 595
- mln::display, 230
- mln::display::impl, 230
- mln::display::impl::generic, 230
- mln::doc, 230
- mln::doc::Accumulator, 595
 - argument, 596
 - init, 596
 - take, 596
- mln::doc::Box, 596
 - bbox, 598
 - bkd_piter, 597
 - fwd_piter, 598
 - has, 598
 - nsites, 598
 - pmax, 599
 - pmin, 599
 - psite, 598
 - site, 598
- mln::doc::Dpoint, 599
 - coord, 600
 - dim, 600
 - dpoint, 600
 - point, 600
- mln::doc::Fastest_Image, 601
 - bbox, 606
 - bkd_piter, 604
 - border, 606
 - buffer, 606
 - coord, 604
 - delta_index, 606
 - domain, 606
 - dpoint, 604
 - fwd_piter, 604
 - has, 606, 607
 - is_valid, 607
 - lvalue, 604
 - nelements, 607
 - nsites, 607
 - operator(), 607, 608
 - point, 604
 - point_at_index, 609
 - pset, 604
 - psite, 605
 - rvalue, 605
 - skeleton, 605
 - value, 605
 - values, 609
 - vset, 605
- mln::doc::Generalized_Pixel, 609
 - ima, 611
 - image, 610
 - rvalue, 610
 - val, 611
 - value, 610
- mln::doc::Image, 611
 - bbox, 615
 - bkd_piter, 613
 - coord, 613
 - domain, 615
 - dpoint, 613
 - fwd_piter, 613
 - has, 615
 - is_valid, 616
 - lvalue, 614
 - nsites, 616
 - operator(), 616
 - point, 614
 - pset, 614
 - psite, 614
 - rvalue, 614
 - skeleton, 614
 - value, 615
 - values, 616
 - vset, 615
- mln::doc::Iterator, 617

- invalidate, 618
- is_valid, 618
- start, 618
- mln::doc::Neighborhood, 618
 - bkd_niter, 619
 - dpoint, 619
 - fwd_niter, 619
 - niter, 619
 - point, 619
- mln::doc::Object, 620
- mln::doc::Pixel_Iterator, 620
 - ima, 622
 - image, 622
 - invalidate, 623
 - is_valid, 623
 - lvalue, 622
 - rvalue, 622
 - start, 623
 - val, 623
 - value, 622
- mln::doc::Point_Site
 - dim, 625
- mln::doc::Point_Site, 623
 - coord, 624
 - dpoint, 624
 - mesh, 624
 - point, 624
 - to_point, 625
- mln::doc::Site_Iterator, 626
 - invalidate, 627
 - is_valid, 627
 - operator psite, 627
 - psite, 627
 - start, 627
- mln::doc::Site_Set, 628
 - bkd_piter, 629
 - fwd_piter, 629
 - has, 629
 - psite, 629
 - site, 629
- mln::doc::Value_Iterator, 630
 - invalidate, 631
 - is_valid, 631
 - operator value, 631
 - start, 631
 - value, 631
- mln::doc::Value_Set, 631
 - bkd_viter, 633
 - fwd_viter, 633
 - has, 633
 - index_of, 633
 - nvalues, 633
 - value, 633
- mln::doc::Weighted_Window, 634
 - bkd_qiter, 635
 - delta, 636
 - dpoint, 635
 - fwd_qiter, 635
 - is_centered, 636
 - is_empty, 636
 - point, 635
 - sym, 636
 - weight, 636
 - win, 636
 - window, 636
- mln::doc::Window, 636
 - bkd_qiter, 637
 - fwd_qiter, 637
 - qiter, 638
- mln::Dpoint, 638
 - to_dpoint, 639
- mln::dpoint, 639
 - coord, 641
 - dim, 642
 - dpoint, 642
 - grid, 641
 - operator mln::algebra::vec< dpoint< G, C
>::dim, Q >, 643
 - psite, 641
 - set_all, 643
 - site, 641
 - to_vec, 643
 - vec, 641
- mln::dpoints_bkd_pixter, 644
 - center_val, 645
 - dpoints_bkd_pixter, 645
 - invalidate, 645
 - is_valid, 645
 - next, 646
 - start, 646
 - update, 646
- mln::dpoints_fwd_pixter, 646
 - center_val, 648
 - dpoints_fwd_pixter, 647
 - invalidate, 648
 - is_valid, 648
 - next, 648
 - start, 648
 - update, 649
- mln::dpsites_bkd_piter, 649
 - dpsites_bkd_piter, 650
 - next, 650
- mln::dpsites_fwd_piter, 650
 - dpsites_fwd_piter, 651
 - next, 651
- mln::draw, 232
 - box, 232
 - box_plain, 233

- dashed_line, 233
- line, 234
- plot, 234
- polygon, 234
- site_set, 235
- mln::Edge, 652
- mln::edge_image, 652
 - edge_image, 654
 - edge_nbh_t, 653
 - edge_win_t, 653
 - graph_t, 653
 - nbh_t, 653
 - operator(), 654
 - site_function_t, 653
 - skeleton, 654
 - win_t, 654
- mln::estim, 235
 - mean, 236
 - min_max, 236
 - sum, 237
- mln::extended, 654
 - domain, 656
 - extended, 655
 - skeleton, 655
 - value, 655
- mln::extension, 237
 - adjust, 238, 239
 - adjust_duplicate, 239
 - adjust_fill, 239
 - duplicate, 239
 - fill, 239
- mln::extension_fun, 656
 - extension, 658
 - extension_fun, 657
 - has, 658
 - operator(), 658
 - rvalue, 657
 - skeleton, 657
 - value, 657
- mln::extension_ima, 658
 - extension, 660
 - extension_ima, 660
 - has, 660
 - operator(), 660
 - rvalue, 659
 - skeleton, 659
 - value, 660
- mln::extension_val, 661
 - change_extension, 662
 - extension, 662
 - extension_val, 662
 - has, 663
 - operator(), 663
 - rvalue, 662
- skeleton, 662
 - value, 662
- mln::flat_image, 663
 - domain, 665
 - flat_image, 665
 - has, 665
 - lvalue, 664
 - operator(), 665
 - rvalue, 664
 - skeleton, 664
 - value, 664
- mln::fun, 240
- mln::fun::access, 241
- mln::fun::from_accu, 666
- mln::fun::i2v, 242
 - operator<<, 242
- mln::fun::n2v, 242
- mln::fun::n2v::white_gaussian, 666
- mln::fun::p2b, 242
- mln::fun::p2b::antilogy, 667
- mln::fun::p2b::tautology, 668
- mln::fun::p2p, 243
- mln::fun::p2v, 243
- mln::fun::stat, 243
- mln::fun::v2b, 243
- mln::fun::v2b::lnot, 670
- mln::fun::v2b::threshold, 671
- mln::fun::v2i, 244
- mln::fun::v2v, 244
 - f_hsi_to_rgb_3x8, 245
 - f_hsl_to_rgb_3x8, 245
 - f_rgb_to_hsi_f, 245
 - f_rgb_to_hsl_f, 245
- mln::fun::v2v::ch_function_value, 672
- mln::fun::v2v::component, 672
- mln::fun::v2v::l1_norm, 673
- mln::fun::v2v::l2_norm, 674
- mln::fun::v2v::linear, 675
- mln::fun::v2v::linfty_norm, 676
- mln::fun::v2v::rgb8_to_rgbn, 677
 - operator(), 679
- mln::fun::v2w2v, 246
- mln::fun::v2w2v::cos, 679
- mln::fun::v2w_w2v, 246
- mln::fun::v2w_w2v::l1_norm, 680
- mln::fun::v2w_w2v::l2_norm, 682
- mln::fun::v2w_w2v::linfty_norm, 683
- mln::fun::vv2b, 246
- mln::fun::vv2b::eq, 684
- mln::fun::vv2b::ge, 684
- mln::fun::vv2b::gt, 685
- mln::fun::vv2b::implies, 686
- mln::fun::vv2b::le, 687
- mln::fun::vv2b::lt, 688

- mln::fun::vv2v, 247
- mln::fun::vv2v::diff_abs, 689
- mln::fun::vv2v::land, 690
- mln::fun::vv2v::land_not, 691
- mln::fun::vv2v::lor, 692
- mln::fun::vv2v::lxor, 693
- mln::fun::vv2v::max, 694
- mln::fun::vv2v::min, 695
- mln::fun::vv2v::vec, 696
- mln::fun::x2p, 248
- mln::fun::x2p::closest_point, 697
- mln::fun::x2v, 248
- mln::fun::x2v::bilinear, 698
 - operator(), 698
- mln::fun::x2v::trilinear, 699
- mln::fun::x2x, 248
- mln::fun::x2x::composed, 699
 - composed, 700
- mln::fun::x2x::linear, 700
 - ima, 701
 - linear, 701
 - operator(), 701
- mln::fun::x2x::rotation, 701
 - data_t, 703
 - inv, 704
 - invert, 703
 - operator(), 704
 - rotation, 703, 704
 - set_alpha, 704
 - set_axis, 704
- mln::fun::x2x::translation, 705
 - data_t, 706
 - inv, 707
 - invert, 706
 - operator(), 707
 - set_t, 707
 - t, 707
 - translation, 706
- mln::fun_image, 707
 - fun_image, 709
 - lvalue, 708
 - operator(), 709
 - rvalue, 708
 - skeleton, 708
 - value, 709
- mln::Function, 710
 - Function, 710
- mln::Function< void >, 710
- mln::Function_n2v, 711
- mln::Function_v2b, 711
- mln::Function_v2v, 712
- mln::Function_vv2b, 713
- mln::Function_vv2v, 713
- mln::fwd_pixter1d, 714
 - fwd_pixter1d, 715
 - image, 715
 - next, 715
- mln::fwd_pixter2d, 715
 - fwd_pixter2d, 716
 - image, 716
 - next, 717
- mln::fwd_pixter3d, 717
 - fwd_pixter3d, 718
 - image, 717
 - next, 718
- mln::Gdpoint, 718
- mln::Gdpoint< void >, 719
- mln::Generalized_Pixel, 720
- mln::geom, 249
 - bbox, 253, 254
 - chamfer, 254
 - delta, 254
 - max_col, 254, 255
 - max_ind, 255
 - max_row, 255
 - max_sli, 255
 - mesh_corner_point_area, 255
 - mesh_curvature, 256
 - mesh_normal, 256
 - min_col, 256, 257
 - min_ind, 257
 - min_row, 257
 - min_sli, 257
 - ncols, 257, 258
 - ninds, 258
 - nrows, 258
 - nsites, 258
 - nslis, 258
 - pmin_pmax, 259
 - rotate, 259, 260
 - seeds2tiling, 260
 - seeds2tiling_roundness, 261
 - translate, 261, 262
 - vertical_symmetry, 262
- mln::geom::complex_geometry, 720
 - add_location, 721
 - complex_geometry, 721
 - operator(), 721
- mln::geom::impl, 262
 - seeds2tiling, 263
- mln::Gpoint, 722
 - operator<<, 725
 - operator+, 723
 - operator+==, 724
 - operator-, 724
 - operator-=, 725
 - operator/, 725
 - operator==, 725

- mln::Graph, 726
- mln::graph, 263
 - compute, 264
 - labeling, 264
 - to_neighb, 264
 - to_win, 265
- mln::graph::attribute::card_t, 727
 - result, 727
- mln::graph::attribute::representative_t, 727
 - result, 728
- mln::graph_elt_mixed_neighborhood, 728
 - bkd_niter, 729
 - fwd_niter, 729
 - niter, 729
- mln::graph_elt_mixed_window, 729
 - bkd_qiter, 731
 - center_t, 731
 - delta, 732
 - fwd_qiter, 731
 - graph_element, 732
 - is_centered, 732
 - is_empty, 733
 - is_symmetric, 733
 - is_valid, 733
 - psite, 732
 - qiter, 732
 - site, 732
 - sym, 733
 - target, 732
- mln::graph_elt_neighborhood, 733
 - bkd_niter, 735
 - fwd_niter, 735
 - niter, 735
- mln::graph_elt_neighborhood_if, 735
 - bkd_niter, 736
 - fwd_niter, 736
 - graph_elt_neighborhood_if, 737
 - mask, 737
 - niter, 736
- mln::graph_elt_window, 737
 - bkd_qiter, 739
 - center_t, 739
 - delta, 740
 - fwd_qiter, 739
 - graph_element, 740
 - is_centered, 740
 - is_empty, 741
 - is_symmetric, 741
 - is_valid, 741
 - psite, 740
 - qiter, 740
 - site, 740
 - sym, 741
 - target, 740
- mln::graph_elt_window_if, 741
 - bkd_qiter, 743
 - change_mask, 745
 - delta, 745
 - fwd_qiter, 743
 - graph_elt_window_if, 745
 - is_centered, 745
 - is_empty, 745
 - is_symmetric, 745
 - is_valid, 746
 - mask, 746
 - mask_t, 744
 - psite, 744
 - qiter, 744
 - site, 744
 - sym, 746
 - target, 744
- mln::graph_window_base, 746
 - delta, 747
 - is_centered, 747
 - is_empty, 747
 - is_symmetric, 748
 - is_valid, 748
 - site, 747
 - sym, 748
- mln::graph_window_if_piter, 748
 - element, 749
 - graph_window_if_piter, 749
 - id, 749
 - next, 750
 - P, 749
- mln::graph_window_piter, 750
 - center_t, 751
 - change_target_site_set, 752
 - element, 753
 - graph_element, 751
 - graph_window_piter, 752
 - id, 753
 - next, 753
 - P, 751
 - target_site_set, 753
- mln::grid, 265
- mln::hexa, 753
 - bkd_piter, 755
 - domain, 756
 - fwd_piter, 755
 - has, 756
 - hexa, 756
 - lvalue, 755
 - operator(), 757
 - psite, 755
 - rvalue, 756
 - skeleton, 756
 - value, 756

- mln::histo, 266
 - compute, 266
 - equalize, 266
- mln::histo::array, 757
- mln::histo::impl, 267
- mln::histo::impl::generic, 267
- mln::Image, 757
- mln::image1d, 759
 - bbox, 762
 - border, 762
 - buffer, 762
 - delta_index, 762
 - domain, 762
 - element, 762
 - has, 763
 - image1d, 761
 - lvalue, 761
 - nelements, 763
 - ninds, 763
 - operator(), 763
 - point_at_index, 763
 - rvalue, 761
 - skeleton, 761
 - value, 761
 - vbbox, 764
- mln::image2d, 764
 - bbox, 767
 - border, 767
 - buffer, 767
 - delta_index, 767
 - domain, 767
 - element, 767
 - has, 768
 - image2d, 766
 - lvalue, 766
 - ncols, 768
 - nelements, 768
 - nrows, 768
 - operator(), 768
 - point_at_index, 768
 - rvalue, 766
 - skeleton, 766
 - value, 766
- mln::image2d_h, 769
 - bkd_piter, 770
 - domain, 771
 - fwd_piter, 770
 - has, 771
 - image2d_h, 771
 - lvalue, 770
 - operator(), 772
 - psite, 771
 - rvalue, 771
 - skeleton, 771
 - value, 771
- mln::image3d, 772
 - bbox, 775
 - border, 775
 - buffer, 775
 - delta_index, 775
 - domain, 776
 - element, 776
 - has, 776
 - image3d, 775
 - lvalue, 774
 - ncols, 776
 - nelements, 776
 - nrows, 777
 - nslis, 777
 - operator(), 777
 - point_at_index, 777
 - rvalue, 774
 - skeleton, 774
 - value, 774
 - vbbox, 777
- mln::impl, 267
- mln::interpolated, 777
 - has, 779
 - interpolated, 779
 - is_valid, 779
 - lvalue, 778
 - psite, 778
 - rvalue, 779
 - skeleton, 779
 - value, 779
- mln::io, 267
- mln::io::cloud, 269
 - load, 269
 - save, 269
- mln::io::dicom, 270
 - get_header, 270
 - load, 270
- mln::io::dicom::dicom_header, 780
- mln::io::dump, 271
 - get_header, 271
 - load, 271
 - save, 272
- mln::io::dump::dump_header, 780
- mln::io::fits, 272
 - load, 272, 273
- mln::io::fld, 273
 - load, 274
 - read_header, 274
 - write_header, 274
- mln::io::fld::fld_header, 780
- mln::io::magick, 274
 - load, 275
 - save, 275

- mln::io::off, 276
 - load, 276
 - save, 277
 - save_bin_alt, 277
- mln::io::pbm, 277
 - load, 278
 - save, 278
- mln::io::pbm::impl, 279
- mln::io::pbms, 279
 - load, 279
- mln::io::pbms::impl, 280
- mln::io::pfm, 280
 - load, 281
 - save, 281
- mln::io::pfm::impl, 281
- mln::io::pgm, 282
 - load, 282
 - save, 282
- mln::io::pgms, 283
 - load, 283
- mln::io::plot, 283
 - load, 284
 - save, 284, 285
- mln::io::pnm, 285
 - load, 286
 - load_raw_2d, 286
 - max_component, 286
 - save, 286
- mln::io::pnm::impl, 287
- mln::io::pnms, 287
 - load, 287, 288
- mln::io::ppm, 288
 - load, 288, 289
 - save, 289
- mln::io::ppms, 289
 - load, 290
- mln::io::raw, 290
 - get_header, 291
 - load, 291
 - save, 291
- mln::io::raw::raw_header, 780
- mln::io::tiff, 291
 - load, 292
- mln::io::txt, 292
 - save, 292
- mln::Iterator, 781
 - next, 783
- mln::labeled_image, 783
 - bbox, 786
 - bbox_t, 785
 - bboxes, 786
 - labeled_image, 786
 - nlabels, 786
 - relabel, 786
 - skeleton, 785
 - subdomain, 787
 - update_data, 787
- mln::labeled_image_base, 787
 - bbox, 789
 - bbox_t, 789
 - bboxes, 789
 - labeled_image_base, 789
 - nlabels, 789
 - relabel, 789, 790
 - subdomain, 790
 - update_data, 790
- mln::labeling, 292
 - background, 296
 - blobs, 296
 - blobs_and_compute, 296
 - colorize, 297, 298
 - compute, 298, 299
 - compute_image, 300
 - fill_holes, 301
 - flat_zones, 301
 - foreground, 302
 - pack, 302, 303
 - pack_inplace, 303
 - regional_maxima, 303
 - regional_minima, 304
 - relabel, 304, 305
 - relabel_inplace, 305
 - superpose, 306
 - value, 306
 - value_and_compute, 306
 - wrap, 307
- mln::labeling::impl, 308
 - compute_fastest, 308, 309
- mln::labeling::impl::generic, 309
 - compute, 310, 311
- mln::lazy_image, 790
 - domain, 792
 - has, 792
 - lazy_image, 792
 - lvalue, 792
 - operator(), 792, 793
 - rvalue, 792
 - skeleton, 792
- mln::linear, 311
 - gaussian, 312, 313
 - gaussian_1st_derivative, 313
 - gaussian_2nd_derivative, 313, 314
 - mln_ch_convolve, 314, 315
 - mln_ch_convolve_grad, 315
- mln::linear::impl, 315
- mln::linear::local, 315
 - convolve, 316
- mln::linear::local::impl, 317

- mln::Literal, 793
- mln::literal, 317
 - black, 320
 - blue, 320
 - brown, 320
 - cyan, 320
 - dark_gray, 320
 - green, 320
 - identity, 321
 - light_gray, 321
 - lime, 321
 - magenta, 321
 - max, 321
 - medium_gray, 321
 - min, 321
 - olive, 321
 - one, 322
 - orange, 322
 - origin, 322
 - pink, 322
 - purple, 322
 - red, 322
 - teal, 322
 - violet, 322
 - white, 323
 - yellow, 323
 - zero, 323
- mln::literal::black_t, 795
- mln::literal::blue_t, 796
- mln::literal::brown_t, 796
- mln::literal::cyan_t, 797
- mln::literal::green_t, 798
- mln::literal::identity_t, 799
- mln::literal::light_gray_t, 800
- mln::literal::lime_t, 801
- mln::literal::magenta_t, 802
- mln::literal::max_t, 803
- mln::literal::min_t, 804
- mln::literal::olive_t, 805
- mln::literal::one_t, 806
- mln::literal::orange_t, 807
- mln::literal::origin_t, 808
- mln::literal::pink_t, 809
- mln::literal::purple_t, 810
- mln::literal::red_t, 811
- mln::literal::teal_t, 812
- mln::literal::violet_t, 813
- mln::literal::white_t, 814
- mln::literal::yellow_t, 815
- mln::literal::zero_t, 816
- mln::logical, 323
 - and_inplace, 324
 - and_not, 324
 - and_not_inplace, 325
 - not_inplace, 325
 - or_inplace, 325
 - xor_inplace, 326
- mln::logical::impl, 326
- mln::logical::impl::generic, 327
- mln::make, 327
 - attachment, 332
 - box1d, 332
 - box2d, 333
 - box2d_h, 334
 - box3d, 334, 335
 - cell, 335
 - couple, 336
 - detachment, 336
 - dpoint2d_h, 336
 - dummy_p_edges, 337
 - dummy_p_vertices, 337, 338
 - edge_image, 338, 339
 - h_mat, 340
 - image, 340, 341
 - image2d, 341
 - image3d, 341
 - influence_zone_adjacency_graph, 341
 - mat, 342
 - ord_pair, 342
 - p_edges_with_mass_centers, 342
 - p_vertices_with_mass_centers, 343
 - pix, 343
 - pixel, 343
 - point2d_h, 344
 - rag_and_labeled_wsl, 344
 - region_adjacency_graph, 345
 - relabelfun, 345
 - vec, 346, 347
 - vertex_image, 347
 - voronoi, 348
 - w_window, 348
 - w_window1d, 348
 - w_window2d, 349
 - w_window3d, 349
 - w_window_directional, 349
- mln::math, 350
 - abs, 350, 351
- mln::Mesh, 817
- mln::Meta_Accumulator, 818
- mln::Meta_Function, 820
- mln::Meta_Function_v2v, 821
- mln::Meta_Function_vv2v, 821
- mln::metal, 351
- mln::metal::ands, 822
- mln::metal::converts_to, 822
- mln::metal::equal, 822
- mln::metal::goes_to, 823
- mln::metal::impl, 352

- mln::metal::is, 823
- mln::metal::is_a, 824
- mln::metal::is_not, 824
- mln::metal::is_not_a, 824
- mln::metal::math, 352
- mln::metal::math::impl, 352
- mln::morpho, 353
 - complementation, 356
 - complementation_inplace, 356
 - contrast, 356
 - dilation, 356
 - erosion, 356
 - erosion_tolerant, 357
 - general, 357
 - gradient, 357
 - gradient_external, 357
 - gradient_internal, 357
 - hit_or_miss, 358
 - hit_or_miss_background_closing, 358
 - hit_or_miss_background_opening, 358
 - hit_or_miss_closing, 358
 - hit_or_miss_opening, 359
 - laplacian, 359
 - line_gradient, 359
 - meyer_wst, 359, 360
 - min, 360
 - min_inplace, 360
 - minus, 360
 - plus, 360
 - rank_filter, 361
 - thick_miss, 361
 - thickening, 361
 - thin_fit, 361
 - thinning, 361
 - top_hat_black, 362
 - top_hat_self_complementary, 362
 - top_hat_white, 362
- mln::morpho::approx, 363
- mln::morpho::attribute, 363
- mln::morpho::attribute::card, 824
 - init, 825
 - is_valid, 825
 - take_as_init, 825
 - take_n_times, 825
 - to_result, 826
- mln::morpho::attribute::count_adjacent_vertices, 826
 - init, 827
 - is_valid, 827
 - take_as_init, 827
 - take_n_times, 827
 - to_result, 827
- mln::morpho::attribute::height, 827
 - base_level, 828
 - init, 828
 - is_valid, 828
 - take_as_init, 828
 - take_n_times, 829
 - to_result, 829
- mln::morpho::attribute::sharpness, 829
 - area, 830
 - height, 830
 - init, 830
 - is_valid, 830
 - take_as_init, 830
 - take_n_times, 831
 - to_result, 831
 - volume, 831
- mln::morpho::attribute::sum, 831
 - init, 832
 - is_valid, 832
 - set_value, 832
 - take_as_init, 832
 - take_n_times, 832
 - to_result, 833
 - untake, 833
- mln::morpho::attribute::volume, 833
 - area, 834
 - init, 834
 - is_valid, 834
 - take_as_init, 834
 - take_n_times, 834
 - to_result, 834
- mln::morpho::closing::approx, 363
 - structural, 364
- mln::morpho::elementary, 364
 - closing, 365
 - mln_trait_op_minus_twice, 365
 - opening, 365
 - top_hat_black, 365
 - top_hat_self_complementary, 365
 - top_hat_white, 366
- mln::morpho::impl, 366
- mln::morpho::impl::generic, 366
- mln::morpho::opening::approx, 367
 - structural, 367
- mln::morpho::reconstruction, 367
- mln::morpho::reconstruction::by_dilation, 368
- mln::morpho::reconstruction::by_erosion, 368
- mln::morpho::tree, 368
 - compute_attribute_image, 370
 - compute_attribute_image_from, 370
 - compute_parent, 371
 - dual_input_max_tree, 372
 - max_tree, 372
 - min_tree, 372
 - propagate_if, 373
 - propagate_if_value, 373

- propagate_node_to_ancestors, 374
- propagate_node_to_descendants, 374, 375
- propagate_representative, 375
- mln::morpho::tree::filter, 375
 - direct, 376
 - filter, 376
 - max, 377
 - min, 377
 - subtractive, 377
- mln::morpho::watershed, 378
 - flooding, 379
 - superpose, 379
 - topological, 380
- mln::morpho::watershed::watershed, 380
- mln::morpho::watershed::watershed::generic, 380
- mln::neighb, 835
 - bkd_niter, 835
 - fwd_niter, 836
 - neighb, 836
 - niter, 836
- mln::Neighborhood, 836
- mln::Neighborhood< void >, 837
- mln::norm, 381
 - l1, 382
 - l1_distance, 382
 - l2, 382
 - l2_distance, 382
 - linfty, 382
 - linfty_distance, 382
 - sqr_l2, 382
- mln::norm::impl, 383
- mln::Object, 837
- mln::opt, 383
 - at, 384
- mln::opt::impl, 385
- mln::p2p_image, 837
 - domain, 839
 - fun, 839
 - operator(), 839
 - p2p_image, 839
 - skeleton, 838
- mln::p_array, 839
 - append, 842
 - bkd_piter, 841
 - change, 843
 - clear, 843
 - element, 841
 - fwd_piter, 841
 - has, 843
 - i_element, 842
 - insert, 843
 - is_valid, 843
 - memory_size, 843
 - nsites, 844
 - p_array, 842
 - piter, 842
 - psite, 842
 - reserve, 844
 - resize, 844
 - std_vector, 845
- mln::p_centered, 845
 - bkd_piter, 846
 - center, 847
 - element, 846
 - fwd_piter, 846
 - has, 847
 - is_valid, 847
 - memory_size, 848
 - p_centered, 847
 - piter, 846
 - psite, 847
 - site, 847
 - window, 848
- mln::p_complex, 848
 - bkd_piter, 849
 - cplx, 850, 851
 - element, 849
 - fwd_piter, 850
 - geom, 851
 - has, 851
 - is_valid, 851
 - nfaces, 851
 - nfaces_of_dim, 851
 - nsites, 852
 - p_complex, 850
 - piter, 850
 - psite, 850
- mln::p_edges, 852
 - bkd_piter, 854
 - edge, 854
 - element, 854
 - fun_t, 854
 - function, 856
 - fwd_piter, 854
 - graph, 856
 - graph_element, 854
 - graph_t, 855
 - has, 856, 857
 - invalidate, 857
 - is_valid, 857
 - memory_size, 857
 - nedges, 857
 - nsites, 857
 - p_edges, 855, 856
 - piter, 855
 - psite, 855
- mln::p_faces, 858
 - bkd_piter, 859

- cplx, 860
- element, 859
- fwd_piter, 859
- is_valid, 860
- nfaces, 860
- nsites, 861
- p_faces, 859, 860
- piter, 859
- psite, 859
- mln::p_graph_piter, 861
 - graph, 862
 - id, 862
 - mln_q_subject, 862
 - next, 862
 - p_graph_piter, 862
- mln::p_if, 862
 - bkd_piter, 864
 - element, 864
 - fwd_piter, 864
 - has, 865
 - is_valid, 865
 - memory_size, 865
 - overset, 865
 - p_if, 864
 - piter, 864
 - pred, 865
 - predicate, 865
 - psite, 864
- mln::p_image, 866
 - bkd_piter, 867
 - clear, 868
 - element, 867
 - fwd_piter, 867
 - has, 869
 - i_element, 867
 - insert, 869
 - is_valid, 869
 - memory_size, 869
 - nsites, 869
 - operator typename internal::p_image_site_-
set< I >::ret, 869
 - p_image, 868
 - piter, 868
 - psite, 868
 - r_element, 868
 - remove, 869
 - S, 868
 - toggle, 870
- mln::p_indexed_bkd_piter, 870
 - index, 871
 - next, 871
 - p_indexed_bkd_piter, 871
- mln::p_indexed_fwd_piter, 871
 - index, 872
 - next, 872
 - p_indexed_fwd_piter, 872
- mln::p_indexed_psite, 873
 - bkd_piter, 875
 - change_key, 876
 - change_keys, 876
 - clear, 876
 - element, 875
 - exists_key, 876
 - fwd_piter, 875
 - has, 877
 - i_element, 875
 - insert, 877
 - is_valid, 877
 - key, 877
 - keys, 877
 - memory_size, 877
 - nsites, 878
 - operator(), 878
 - p_key, 876
 - piter, 875
 - psite, 875
 - r_element, 876
 - remove, 878
 - remove_key, 878
- mln::p_line2d, 878
 - bbox, 881
 - begin, 881
 - bkd_piter, 880
 - element, 880
 - end, 881
 - fwd_piter, 880
 - has, 881
 - is_valid, 882
 - memory_size, 882
 - nsites, 882
 - p_line2d, 881
 - piter, 880
 - psite, 880
 - q_box, 880
 - std_vector, 882
- mln::p_mutable_array_of, 882
 - bkd_piter, 884
 - clear, 885
 - element, 884
 - fwd_piter, 884
 - has, 885
 - i_element, 884
 - insert, 885
 - is_valid, 885
 - memory_size, 885
 - nelements, 885
 - p_mutable_array_of, 885

- ptiter, [884](#)
 - psite, [884](#)
 - reserve, [886](#)
- mln::p_n_faces_bkd_ptiter, [886](#)
 - n, [887](#)
 - next, [887](#)
 - p_n_faces_bkd_ptiter, [887](#)
- mln::p_n_faces_fwd_ptiter, [887](#)
 - n, [888](#)
 - next, [888](#)
 - p_n_faces_fwd_ptiter, [888](#)
- mln::p_priority, [889](#)
 - bkd_ptiter, [890](#)
 - clear, [891](#)
 - element, [891](#)
 - exists_priority, [892](#)
 - front, [892](#)
 - fwd_ptiter, [891](#)
 - has, [892](#)
 - highest_priority, [892](#)
 - i_element, [891](#)
 - insert, [892](#)
 - is_valid, [893](#)
 - lowest_priority, [893](#)
 - memory_size, [893](#)
 - nsites, [893](#)
 - operator(), [893](#)
 - p_priority, [891](#)
 - ptiter, [891](#)
 - pop, [893](#)
 - pop_front, [894](#)
 - priorities, [894](#)
 - psite, [891](#)
 - push, [894](#)
- mln::p_queue, [894](#)
 - bkd_ptiter, [896](#)
 - clear, [897](#)
 - element, [896](#)
 - front, [897](#)
 - fwd_ptiter, [896](#)
 - has, [897](#)
 - i_element, [896](#)
 - insert, [897](#)
 - is_valid, [898](#)
 - memory_size, [898](#)
 - nsites, [898](#)
 - p_queue, [897](#)
 - ptiter, [896](#)
 - pop, [898](#)
 - pop_front, [898](#)
 - psite, [897](#)
 - push, [899](#)
 - std_deque, [899](#)
- mln::p_queue_fast, [899](#)
 - bkd_ptiter, [901](#)
 - clear, [902](#)
 - compute_has, [902](#)
 - element, [901](#)
 - empty, [902](#)
 - front, [902](#)
 - fwd_ptiter, [901](#)
 - has, [902](#)
 - i_element, [901](#)
 - insert, [903](#)
 - is_valid, [903](#)
 - memory_size, [903](#)
 - nsites, [903](#)
 - p_queue_fast, [902](#)
 - ptiter, [901](#)
 - pop, [903](#)
 - pop_front, [904](#)
 - psite, [901](#)
 - purge, [904](#)
 - push, [904](#)
 - reserve, [904](#)
 - std_vector, [904](#)
- mln::p_run, [904](#)
 - bbox, [907](#)
 - bkd_ptiter, [906](#)
 - element, [906](#)
 - end, [907](#)
 - fwd_ptiter, [906](#)
 - has, [908](#)
 - has_index, [908](#)
 - init, [908](#)
 - is_valid, [908](#)
 - length, [908](#)
 - memory_size, [908](#)
 - nsites, [909](#)
 - p_run, [907](#)
 - ptiter, [906](#)
 - psite, [906](#)
 - q_box, [907](#)
 - start, [909](#)
- mln::p_set, [909](#)
 - bkd_ptiter, [911](#)
 - clear, [912](#)
 - element, [911](#)
 - fwd_ptiter, [911](#)
 - has, [912](#)
 - i_element, [911](#)
 - insert, [912](#)
 - is_valid, [912](#)
 - memory_size, [912](#)
 - nsites, [913](#)
 - p_set, [912](#)
 - ptiter, [911](#)
 - psite, [911](#)

- r_element, 911
 - remove, 913
 - std_vector, 913
 - util_set, 913
- mln::p_transformed, 913
 - bkd_piter, 915
 - element, 915
 - function, 916
 - fwd_piter, 915
 - has, 916
 - is_valid, 916
 - memory_size, 916
 - p_transformed, 915
 - piter, 915
 - primary_set, 916
 - psite, 915
- mln::p_transformed_piter, 916
 - change_target, 917
 - next, 918
 - p_transformed_piter, 917
- mln::p_vaccess, 918
 - bkd_piter, 919
 - element, 920
 - fwd_piter, 920
 - has, 921
 - i_element, 920
 - insert, 921
 - is_valid, 921
 - memory_size, 921
 - operator(), 922
 - p_vaccess, 921
 - piter, 920
 - pset, 920
 - psite, 920
 - value, 920
 - values, 922
 - vset, 920
- mln::p_vertices, 922
 - bkd_piter, 924
 - element, 924
 - fun_t, 924
 - function, 926
 - fwd_piter, 924
 - graph, 926
 - graph_element, 924
 - graph_t, 925
 - has, 927
 - invalidate, 927
 - is_valid, 927
 - memory_size, 927
 - nsites, 927
 - nvertices, 928
 - operator(), 928
 - p_vertices, 925, 926
 - piter, 925
 - psite, 925
 - vertex, 925
- mln::pixel, 928
 - change_to, 929
 - is_valid, 929
 - pixel, 929
- mln::plain, 930
 - operator I, 931
 - operator=, 931
 - plain, 931
 - skeleton, 931
- mln::Point, 932
 - operator+=, 933
 - operator=, 933
 - operator/, 933
 - point, 933
 - to_point, 933
- mln::point, 934
 - coord, 937
 - delta, 937
 - dim, 938
 - dpsite, 937
 - grid, 937
 - h_vec, 937
 - last_coord, 939
 - minus_infty, 939
 - operator+=, 939
 - operator=, 939
 - origin, 941
 - plus_infty, 940
 - point, 938, 939
 - set_all, 940
 - to_h_vec, 940
 - to_vec, 940
 - vec, 938
- mln::Proxy, 941
- mln::Proxy< void >, 941
- mln::Pseudo_Site, 941
- mln::Pseudo_Site< void >, 942
- mln::pw, 385
- mln::pw::image, 942
 - image, 943
 - skeleton, 943
- mln::registration, 385
 - get_rot, 386
 - icp, 386, 387
 - registration1, 387
 - registration2, 388
 - registration3, 388
- mln::registration::closest_point_basic, 943
- mln::registration::closest_point_with_map, 944
- mln::Regular_Grid, 944
- mln::safe_image, 945

- operator safe_image< const I >, 945
- skeleton, 945
- mln::select, 388
- mln::select::p_of, 945
- mln::set, 388
 - card, 389
 - compute, 389
 - compute_with_weights, 390
 - get, 390
 - has, 391
 - mln_meta_accu_result, 391
- mln::Site, 946
- mln::Site< void >, 947
- mln::Site_Iterator, 947
 - next, 949
- mln::Site_Proxy, 949
- mln::Site_Proxy< void >, 950
- mln::Site_Set, 950
 - diff, 952
 - inter, 952
 - operator<, 952
 - operator<=, 952
 - operator<=, 952
 - operator==, 952
 - sym_diff, 953
 - uni, 953
 - unique, 953
- mln::Site_Set< void >, 953
- mln::sub_image, 953
 - domain, 955
 - operator sub_image< const I, S >, 955
 - skeleton, 954
 - sub_image, 954
- mln::sub_image_if, 955
 - domain, 956
 - skeleton, 956
 - sub_image_if, 956
- mln::subsampling, 391
 - antialiased, 392
 - gaussian_subsampling, 392
 - subsampling, 392
- mln::tag, 393
- mln::test, 393
 - positive, 394
 - predicate, 394
- mln::test::impl, 395
- mln::thru_image, 956
 - operator thru_image< const I, F >, 957
- mln::thru_image, 957
 - operator thru_image< const I1, const I2, F >, 958
 - psite, 958
 - rvalue, 958
 - skeleton, 958
 - value, 958
- mln::topo, 395
 - detach, 399
 - edge, 400
 - is_facet, 400
 - make_algebraic_face, 400
 - make_algebraic_n_face, 400
 - operator<, 402, 403
 - operator<=, 403, 404
 - operator+, 401
 - operator-, 402
 - operator==, 404, 405
- mln::topo::adj_higher_dim_connected_n_face_-bkd_iter, 959
 - adj_higher_dim_connected_n_face_bkd_iter, 959
 - next, 959
- mln::topo::adj_higher_dim_connected_n_face_-fwd_iter, 960
 - adj_higher_dim_connected_n_face_fwd_iter, 960
 - next, 961
- mln::topo::adj_higher_face_bkd_iter, 961
 - adj_higher_face_bkd_iter, 962
 - next, 962
- mln::topo::adj_higher_face_fwd_iter, 962
 - adj_higher_face_fwd_iter, 963
 - next, 963
- mln::topo::adj_lower_dim_connected_n_face_-bkd_iter, 963
 - adj_lower_dim_connected_n_face_bkd_iter, 964
 - next, 964
- mln::topo::adj_lower_dim_connected_n_face_-fwd_iter, 964
 - adj_lower_dim_connected_n_face_fwd_iter, 965
 - next, 965
- mln::topo::adj_lower_face_bkd_iter, 965
 - adj_lower_face_bkd_iter, 966
 - next, 966
- mln::topo::adj_lower_face_fwd_iter, 967
 - adj_lower_face_fwd_iter, 967
 - next, 967
- mln::topo::adj_lower_higher_face_bkd_iter, 968
 - adj_lower_higher_face_bkd_iter, 968
 - next, 968
- mln::topo::adj_lower_higher_face_fwd_iter, 969
 - adj_lower_higher_face_fwd_iter, 969
 - next, 970
- mln::topo::adj_m_face_bkd_iter, 970
 - adj_m_face_bkd_iter, 971
 - next, 971
- mln::topo::adj_m_face_fwd_iter, 971

- adj_m_face_fwd_iter, 972
- next, 972
- mln::topo::algebraic_face, 973
 - algebraic_face, 975
 - cplx, 975
 - data, 975
 - dec_face_id, 975
 - dec_n, 976
 - face_id, 976
 - higher_dim_adj_faces, 976
 - inc_face_id, 976
 - inc_n, 976
 - invalidate, 976
 - is_valid, 976
 - lower_dim_adj_faces, 977
 - n, 977
 - set_cplx, 977
 - set_face_id, 977
 - set_n, 977
 - set_sign, 977
 - sign, 978
- mln::topo::algebraic_n_face, 978
 - algebraic_n_face, 980
 - cplx, 980
 - data, 980
 - dec_face_id, 980
 - face_id, 980
 - higher_dim_adj_faces, 981
 - inc_face_id, 981
 - invalidate, 981
 - is_valid, 981
 - lower_dim_adj_faces, 981
 - n, 982
 - set_cplx, 982
 - set_face_id, 982
 - set_sign, 982
 - sign, 982
- mln::topo::center_only_iter, 982
 - center_only_iter, 983
 - next, 983
- mln::topo::centered_bkd_iter_adapter, 984
 - centered_bkd_iter_adapter, 984
 - next, 984
- mln::topo::centered_fwd_iter_adapter, 985
 - centered_fwd_iter_adapter, 985
 - next, 986
- mln::topo::complex, 986
 - add_face, 988
 - addr, 988
 - bkd_citer, 987
 - complex, 987
 - fwd_citer, 987
 - nfaces, 988
 - nfaces_of_dim, 988
 - nfaces_of_static_dim, 988
 - print, 989
 - print_faces, 989
- mln::topo::face, 989
 - cplx, 991
 - data, 991
 - dec_face_id, 991
 - dec_n, 992
 - face, 991
 - face_id, 992
 - higher_dim_adj_faces, 992
 - inc_face_id, 992
 - inc_n, 992
 - invalidate, 992
 - is_valid, 992
 - lower_dim_adj_faces, 992
 - n, 993
 - set_cplx, 993
 - set_face_id, 993
 - set_n, 993
- mln::topo::face_bkd_iter, 993
 - face_bkd_iter, 994
 - next, 994
 - start, 994
- mln::topo::face_fwd_iter, 995
 - face_fwd_iter, 995
 - next, 995
 - start, 996
- mln::topo::is_n_face, 996
- mln::topo::is_simple_2d_t, 997
- mln::topo::is_simple_cell, 998
 - D, 1001
 - mln_geom, 1000
 - operator(), 1000
 - psite, 1000
 - result, 1000
 - set_image, 1001
- mln::topo::n_face, 1001
 - cplx, 1003
 - data, 1003
 - dec_face_id, 1003
 - face_id, 1003
 - higher_dim_adj_faces, 1003
 - inc_face_id, 1003
 - invalidate, 1004
 - is_valid, 1004
 - lower_dim_adj_faces, 1004
 - n, 1004
 - n_face, 1002
 - set_cplx, 1004
 - set_face_id, 1004
- mln::topo::n_face_bkd_iter, 1005
 - n, 1005
 - next, 1005

- start, 1006
- mln::topo::n_face_fwd_iter, 1006
 - n, 1007
 - next, 1007
 - start, 1007
- mln::topo::n_faces_set, 1007
 - add, 1008
 - faces, 1008
 - faces_type, 1008
 - reserve, 1008
- mln::topo::skeleton::is_simple_point, 1009
- mln::topo::static_n_face_bkd_iter, 1009
 - next, 1010
 - start, 1010
 - static_n_face_bkd_iter, 1010
- mln::topo::static_n_face_fwd_iter, 1011
 - next, 1011
 - start, 1012
 - static_n_face_fwd_iter, 1011
- mln::tr_image, 1012
 - domain, 1014
 - has, 1014
 - is_valid, 1014
 - lvalue, 1013
 - operator(), 1014
 - psite, 1013
 - rvalue, 1013
 - set_tr, 1015
 - site, 1013
 - skeleton, 1014
 - tr, 1015
 - tr_image, 1014
 - value, 1014
- mln::trace, 405
- mln::trait, 405
- mln::transform, 405
 - distance_and_closest_point_geodesic, 407
 - distance_and_influence_zone_geodesic, 408
 - distance_front, 408
 - distance_geodesic, 408
 - hough, 408
 - influence_zone_front, 409
 - influence_zone_geodesic, 409
 - influence_zone_geodesic_saturated, 410
- mln::transformed_image, 1015
 - domain, 1016
 - operator transformed_image< const I, F >, 1016
 - operator(), 1017
 - skeleton, 1016
 - transformed_image, 1016
- mln::unproject_image, 1017
 - domain, 1018
 - operator(), 1018
 - unproject_image, 1018
- mln::util, 410
 - display_branch, 414
 - display_tree, 414
 - lemmings, 414
 - make_greater_point, 415
 - make_greater_psite, 415
 - operator<, 415
 - operator<<, 415
 - operator==, 415, 416
 - ord_strict, 416
 - ord_weak, 416
 - tree_fast_to_image, 416
 - tree_to_fast, 416
 - tree_to_image, 417
 - vertex_id_t, 414
- mln::util::adjacency_matrix, 1019
 - adjacency_matrix, 1019
- mln::util::array, 1019
 - append, 1023
 - array, 1023
 - bkd_eiter, 1022
 - clear, 1023
 - eiter, 1022
 - element, 1022
 - fill, 1024
 - fwd_eiter, 1022
 - is_empty, 1024
 - last, 1024
 - memory_size, 1024
 - nelements, 1025
 - operator(), 1025
 - reserve, 1026
 - resize, 1026
 - result, 1022
 - size, 1026
 - std_vector, 1026
- mln::util::branch, 1027
 - apex, 1028
 - branch, 1027
 - util_tree, 1028
- mln::util::branch_iter, 1028
 - deepness, 1029
 - invalidate, 1029
 - is_valid, 1029
 - next, 1029
 - operator util::tree_node< T > &, 1029
 - start, 1029
- mln::util::branch_iter_ind, 1030
 - deepness, 1030
 - invalidate, 1030
 - is_valid, 1031
 - next, 1031
 - operator util::tree_node< T > &, 1031

- start, 1031
- mln::util::couple, 1031
 - change_both, 1033
 - change_first, 1033
 - change_second, 1033
 - first, 1033
 - second, 1033
- mln::util::eat, 1033
- mln::util::edge, 1034
 - category, 1036
 - change_graph, 1036
 - edge, 1036
 - graph, 1036
 - graph_t, 1036
 - id, 1036
 - id_t, 1036
 - id_value_t, 1036
 - invalidate, 1037
 - is_valid, 1037
 - ith_nbh_edge, 1037
 - nmax_nbh_edges, 1037
 - operator edge_id_t, 1037
 - update_id, 1037
 - v1, 1037
 - v2, 1038
 - v_other, 1038
- mln::util::fibonacci_heap, 1038
 - clear, 1040
 - fibonacci_heap, 1040
 - front, 1040
 - is_empty, 1040
 - is_valid, 1041
 - nelements, 1041
 - operator=, 1041
 - pop_front, 1041
 - push, 1041, 1042
- mln::util::graph, 1042
 - add_edge, 1045
 - add_vertex, 1045
 - add_vertices, 1046
 - e_ith_nbh_edge, 1046
 - e_nmax, 1046
 - edge, 1046
 - edge_fwd_iter, 1044
 - edge_nbh_edge_fwd_iter, 1044
 - edges, 1046
 - edges_set_t, 1044
 - edges_t, 1044
 - graph, 1045
 - has_e, 1047
 - has_v, 1047
 - is_subgraph_of, 1047
 - v1, 1047
 - v2, 1047
 - v_ith_nbh_edge, 1047
 - v_ith_nbh_vertex, 1048
 - v_nmax, 1048
 - vertex, 1048
 - vertex_nbh_edge_fwd_iter, 1044
 - vertex_nbh_vertex_fwd_iter, 1044
 - vertices_t, 1045
- mln::util::greater_point, 1048
 - operator(), 1049
- mln::util::greater_psite, 1049
 - operator(), 1049
- mln::util::head, 1049
- mln::util::ignore, 1050
- mln::util::ilcell, 1050
- mln::util::impl, 417
- mln::util::line_graph, 1051
 - e_ith_nbh_edge, 1053
 - e_nmax, 1054
 - edge, 1054
 - edge_fwd_iter, 1053
 - edge_nbh_edge_fwd_iter, 1053
 - edges_t, 1053
 - graph, 1054
 - has, 1054
 - has_e, 1054
 - has_v, 1055
 - is_subgraph_of, 1055
 - v1, 1055
 - v2, 1055
 - v_ith_nbh_edge, 1055
 - v_ith_nbh_vertex, 1056
 - v_nmax, 1056
 - vertex, 1056
 - vertex_nbh_edge_fwd_iter, 1053
 - vertex_nbh_vertex_fwd_iter, 1053
 - vertices_t, 1053
- mln::util::nil, 1056
- mln::util::node, 1057
- mln::util::object_id, 1057
 - object_id, 1059
 - value_t, 1059
- mln::util::ord, 1059
- mln::util::ord_pair, 1059
 - change_both, 1061
 - change_first, 1061
 - change_second, 1061
 - first, 1061
 - second, 1061
- mln::util::pix, 1062
 - ima, 1063
 - p, 1063
 - pix, 1063
 - psite, 1062
 - v, 1063

- value, 1062
- mln::util::set, 1063
 - bkd_eiter, 1066
 - clear, 1066
 - eiter, 1066
 - element, 1066
 - first_element, 1066
 - fwd_eiter, 1066
 - has, 1067
 - insert, 1067
 - is_empty, 1067
 - last_element, 1068
 - memory_size, 1068
 - nelements, 1068
 - remove, 1069
 - set, 1066
 - std_vector, 1069
- mln::util::site_pair, 1069
 - first, 1070
 - pair, 1070
 - second, 1071
- mln::util::soft_heap, 1071
 - ~soft_heap, 1073
 - clear, 1073
 - element, 1072
 - is_empty, 1073
 - is_valid, 1073
 - nelements, 1073
 - pop_front, 1073
 - push, 1073, 1074
 - soft_heap, 1072
- mln::util::timer, 1074
- mln::util::tracked_ptr, 1075
 - ~tracked_ptr, 1076
 - operator bool, 1076
 - operator->, 1076
 - operator=, 1077
 - tracked_ptr, 1076
- mln::util::tree, 1077
 - add_tree_down, 1078
 - add_tree_up, 1078
 - check_consistency, 1078
 - main_branch, 1079
 - root, 1079
 - tree, 1078
- mln::util::tree_node, 1079
 - add_child, 1081
 - check_consistency, 1081
 - children, 1082
 - delete_tree_node, 1082
 - elt, 1082
 - parent, 1083
 - print, 1083
 - search, 1083
 - search_rec, 1083
 - set_parent, 1084
 - tree_node, 1081
- mln::util::vertex, 1084
 - Category, 1086
 - change_graph, 1087
 - edge_with, 1087
 - graph, 1087
 - graph_t, 1086
 - id, 1087
 - id_t, 1086
 - id_value_t, 1086
 - invalidate, 1087
 - is_valid, 1087
 - ith_nbh_edge, 1087
 - ith_nbh_vertex, 1088
 - nmax_nbh_edges, 1088
 - nmax_nbh_vertices, 1088
 - operator vertex_id_t, 1088
 - other, 1088
 - update_id, 1088
 - vertex, 1086
- mln::util::yes, 1089
- mln::Value, 1089
- mln::value, 417
 - cast, 424
 - equiv, 424
 - float01_16, 422
 - float01_8, 422
 - gl16, 422
 - gl8, 422
 - glf, 422
 - int_s16, 422
 - int_s32, 423
 - int_s8, 423
 - int_u12, 423
 - int_u16, 423
 - int_u32, 423
 - int_u8, 423
 - label_16, 423
 - label_32, 423
 - label_8, 423
 - operator<<, 425–428
 - operator*, 424
 - operator+, 424, 425
 - operator-, 425
 - operator/, 425
 - operator==, 428
 - other, 429
 - rgb16, 424
 - rgb8, 424
 - stack, 429
- mln::value::float01, 1091
 - enc, 1092

- equiv, 1092
- float01, 1092
- nbits, 1092
- operator float, 1092
- set_nbits, 1093
- to_nbits, 1093
- value, 1093
- value_ind, 1093
- mln::value::float01_f, 1093
 - float01_f, 1094
 - operator float, 1094
 - operator=, 1094
 - value, 1094
- mln::value::graylevel, 1094
 - graylevel, 1096
 - operator=, 1097
 - to_float, 1097
 - value, 1097
- mln::value::graylevel_f, 1098
 - graylevel_f, 1099
 - operator graylevel< n >, 1099
 - operator=, 1099, 1100
 - value, 1100
- mln::value::impl, 429
- mln::value::int_s, 1100
 - int_s, 1102
 - one, 1102
 - operator int, 1102
 - operator=, 1102
 - zero, 1102
- mln::value::int_u, 1103
 - int_u, 1104
 - next, 1104
 - operator unsigned, 1104
 - operator-, 1105
 - operator=, 1105
- mln::value::int_u_sat, 1105
 - int_u_sat, 1106
 - one, 1107
 - operator int, 1106
 - operator+=, 1107
 - operator-=, 1107
 - operator=, 1107
 - zero, 1107
- mln::value::Integer, 1107
- mln::value::Integer< void >, 1108
- mln::value::label, 1108
 - enc, 1109
 - label, 1110
 - next, 1110
 - operator unsigned, 1110
 - operator++, 1110
 - operator--, 1110
 - operator=, 1110, 1111
 - prev, 1111
- mln::value::lut_vec, 1111
 - bkd_viter, 1112
 - fwd_viter, 1112
 - has, 1113
 - index_of, 1113
 - lut_vec, 1113
 - nvalues, 1113
 - value, 1112
- mln::value::proxy, 1114
 - ~proxy, 1116
 - enc, 1115
 - equiv, 1115
 - operator=, 1116
 - proxy, 1115
 - to_value, 1116
- mln::value::qt::rgb32, 1116
 - operator=, 1118
 - red, 1118
 - rgb32, 1117
 - zero, 1118
- mln::value::rgb, 1118
 - operator=, 1120
 - red, 1120
 - rgb, 1119
 - zero, 1120
- mln::value::set, 1120
 - the, 1121
- mln::value::sign, 1121
 - enc, 1122
 - equiv, 1122
 - one, 1123
 - operator int, 1123
 - operator=, 1123
 - sign, 1122
 - zero, 1123
- mln::value::stack_image, 1123
 - domain_t, 1124
 - is_valid, 1125
 - lvalue, 1124
 - operator(), 1125, 1126
 - psite, 1124
 - rvalue, 1125
 - skeleton, 1125
 - stack_image, 1125
 - value, 1125
- mln::value::super_value< sign >, 1126
- mln::value::value_array, 1126
 - operator(), 1127
 - value_array, 1127
 - vset, 1127
- mln::Vertex, 1127
- mln::vertex_image, 1128
 - graph_t, 1129

- nbh_t, 1129
- operator(), 1130
- site_function_t, 1129
- skeleton, 1129
- vertex_image, 1129
- win_t, 1129
- mln::violent_cast_image, 1130
 - lvalue, 1131
 - operator(), 1131
 - rvalue, 1131
 - skeleton, 1131
 - value, 1131
 - violent_cast_image, 1131
- mln::w_window, 1132
 - bkd_qiter, 1133
 - clear, 1134
 - dpsite, 1133
 - fwd_qiter, 1133
 - insert, 1134
 - is_symmetric, 1134
 - operator<=, 1135
 - operator==, 1135
 - std_vector, 1134
 - sym, 1134
 - w, 1134
 - w_window, 1134
 - weight, 1133
 - weights, 1135
 - win, 1135
- mln::Weighted_Window, 1135
 - operator-, 1136
- mln::win, 429
 - diff, 431
 - mln_regular, 431
 - sym, 431
- mln::win::backdiag2d, 1137
 - backdiag2d, 1137
 - length, 1138
- mln::win::ball, 1138
 - ball, 1138
 - diameter, 1139
- mln::win::cube3d, 1139
 - cube3d, 1139
 - length, 1140
- mln::win::cuboid3d, 1140
 - cuboid3d, 1141
 - depth, 1141
 - height, 1141
 - volume, 1141
 - width, 1142
- mln::win::diag2d, 1142
 - diag2d, 1142
 - length, 1143
- mln::win::line, 1143
 - length, 1144
 - line, 1144
 - size, 1144
- mln::win::multiple, 1145
- mln::win::multiple_size, 1145
- mln::win::octagon2d, 1145
 - area, 1146
 - length, 1146
 - octagon2d, 1146
- mln::win::rectangle2d, 1147
 - area, 1148
 - height, 1148
 - rectangle2d, 1147
 - std_vector, 1148
 - width, 1148
- mln::Window, 1148
- mln::window, 1149
 - bkd_qiter, 1151
 - clear, 1152
 - delta, 1152
 - dp, 1152
 - fwd_qiter, 1151
 - has, 1152
 - insert, 1152
 - is_centered, 1153
 - is_empty, 1153
 - is_symmetric, 1153
 - operator==, 1154
 - print, 1153
 - qiter, 1151
 - regular, 1151
 - size, 1153
 - std_vector, 1153
 - sym, 1154
 - window, 1151
- mln::world::inter_pixel::is_separator, 1154
- mln_ch_convolve
 - mln::linear, 314, 315
- mln_ch_convolve_grad
 - mln::linear, 315
- mln_exact
 - mln, 151
- mln_gen_complex_neighborhood
 - mln, 151, 152
- mln_gen_complex_window
 - mln, 152, 153
- mln_gen_complex_window_p
 - mln, 153
- mln_geom
 - mln::topo::is_simple_cell, 1000
- mln_image_from_grid
 - mln::convert, 198, 199
- mln_meta_accu_result
 - mln::accu, 165

- mln::data, 208
- mln::set, 391
- mln_q_subject
 - mln::p_graph_piter, 862
- mln_regular
 - mln, 153
 - mln::win, 431
- mln_trait_op_geq
 - mln, 154
- mln_trait_op_greater
 - mln, 154
- mln_trait_op_leq
 - mln, 154
- mln_trait_op_minus_twice
 - mln::morpho::elementary, 365
- mln_trait_op_neq
 - mln, 154
- mln_window
 - mln::convert, 199
- modneighb1d
 - c2, 102
 - neighb1d, 102
- modneighb2d
 - c2_col, 103
 - c2_row, 103
 - c4, 103
 - c8, 104
 - neighb2d, 103
- modneighb3d
 - c18, 105
 - c26, 105
 - c2_3d_sli, 106
 - c4_3d, 106
 - c6, 107
 - c8_3d, 107
 - neighb3d, 105
- modwin1d
 - segment1d, 112
 - window1d, 113
- modwin2d
 - disk2d, 114
 - hline2d, 114
 - vline2d, 114
 - win_c4p, 115
 - win_c8p, 115
 - window2d, 114
- modwin3d
 - sline3d, 116
 - sphere3d, 116
 - win_c4p_3d, 117
 - win_c8p_3d, 117
 - window3d, 117
- mosaic
 - mln::debug, 226

Multiple accumulators, 94

Multiple windows, 118

n

- mln::complex_psite, 587
- mln::p_n_faces_bkd_piter, 887
- mln::p_n_faces_fwd_piter, 888
- mln::topo::algebraic_face, 977
- mln::topo::algebraic_n_face, 982
- mln::topo::face, 993
- mln::topo::n_face, 1004
- mln::topo::n_face_bkd_iter, 1005
- mln::topo::n_face_fwd_iter, 1007

N-D windows, 118

n_face

- mln::topo::n_face, 1002

n_items

- mln::accu::stat::var, 532
- mln::accu::stat::variance, 534

nbh_t

- mln::edge_image, 653
- mln::vertex_image, 1129

nbits

- mln::value::float01, 1092

ncols

- mln::geom, 257, 258
- mln::image2d, 768
- mln::image3d, 776

nedges

- mln::p_edges, 857

neighb

- mln::neighb, 836

neighb1d

- modneighb1d, 102

neighb2d

- modneighb2d, 103

neighb3d

- modneighb3d, 105

Neighborhoods, 101

nelements

- mln::doc::Fastest_Image, 607
- mln::image1d, 763
- mln::image2d, 768
- mln::image3d, 776
- mln::p_mutable_array_of, 885
- mln::util::array, 1025
- mln::util::fibonacci_heap, 1041
- mln::util::set, 1068
- mln::util::soft_heap, 1073

next

- mln::bkd_pixter1d, 545
- mln::bkd_pixter2d, 546
- mln::bkd_pixter3d, 548
- mln::box_runend_piter, 562

- mln::box_runstart_piter, 563
- mln::complex_neighborhood_bkd_piter, 583
- mln::complex_neighborhood_fwd_piter, 585
- mln::complex_window_bkd_piter, 589
- mln::complex_window_fwd_piter, 591
- mln::dpoints_bkd_pixter, 646
- mln::dpoints_fwd_pixter, 648
- mln::dpsites_bkd_piter, 650
- mln::dpsites_fwd_piter, 651
- mln::fwd_pixter1d, 715
- mln::fwd_pixter2d, 717
- mln::fwd_pixter3d, 718
- mln::graph_window_if_piter, 750
- mln::graph_window_piter, 753
- mln::Iterator, 783
- mln::p_graph_piter, 862
- mln::p_indexed_bkd_piter, 871
- mln::p_indexed_fwd_piter, 872
- mln::p_n_faces_bkd_piter, 887
- mln::p_n_faces_fwd_piter, 888
- mln::p_transformed_piter, 918
- mln::Site_Iterator, 949
- mln::topo::adj_higher_dim_connected_n_-
face_bkd_iter, 959
- mln::topo::adj_higher_dim_connected_n_-
face_fwd_iter, 961
- mln::topo::adj_higher_face_bkd_iter, 962
- mln::topo::adj_higher_face_fwd_iter, 963
- mln::topo::adj_lower_dim_connected_n_-
face_bkd_iter, 964
- mln::topo::adj_lower_dim_connected_n_-
face_fwd_iter, 965
- mln::topo::adj_lower_face_bkd_iter, 966
- mln::topo::adj_lower_face_fwd_iter, 967
- mln::topo::adj_lower_higher_face_bkd_iter,
968
- mln::topo::adj_lower_higher_face_fwd_iter,
970
- mln::topo::adj_m_face_bkd_iter, 971
- mln::topo::adj_m_face_fwd_iter, 972
- mln::topo::center_only_iter, 983
- mln::topo::centered_bkd_iter_adapter, 984
- mln::topo::centered_fwd_iter_adapter, 986
- mln::topo::face_bkd_iter, 994
- mln::topo::face_fwd_iter, 995
- mln::topo::n_face_bkd_iter, 1005
- mln::topo::n_face_fwd_iter, 1007
- mln::topo::static_n_face_bkd_iter, 1010
- mln::topo::static_n_face_fwd_iter, 1011
- mln::util::branch_iter, 1029
- mln::util::branch_iter_ind, 1031
- mln::value::int_u, 1104
- mln::value::label, 1110
- nfaces
- mln::p_complex, 851
- mln::p_faces, 860
- mln::topo::complex, 988
- nfaces_of_dim
 - mln::p_complex, 851
 - mln::topo::complex, 988
- nfaces_of_static_dim
 - mln::topo::complex, 988
- ninds
 - mln::geom, 258
 - mln::image1d, 763
- niter
 - mln::doc::Neighborhood, 619
 - mln::graph_elt_mixed_neighborhood, 729
 - mln::graph_elt_neighborhood, 735
 - mln::graph_elt_neighborhood_if, 736
 - mln::neighb, 836
- nlabels
 - mln::labeled_image, 786
 - mln::labeled_image_base, 789
- nmax_nbh_edges
 - mln::util::edge, 1037
 - mln::util::vertex, 1088
- nmax_nbh_vertices
 - mln::util::vertex, 1088
- not_inplace
 - mln::logical, 325
- nrows
 - mln::geom, 258
 - mln::image2d, 768
 - mln::image3d, 777
- nsites
 - mln::accu::center, 434
 - mln::Box, 559
 - mln::box, 554
 - mln::doc::Box, 598
 - mln::doc::Fastest_Image, 607
 - mln::doc::Image, 616
 - mln::geom, 258
 - mln::p_array, 844
 - mln::p_complex, 852
 - mln::p_edges, 857
 - mln::p_faces, 861
 - mln::p_image, 869
 - mln::p_key, 878
 - mln::p_line2d, 882
 - mln::p_priority, 893
 - mln::p_queue, 898
 - mln::p_queue_fast, 903
 - mln::p_run, 909
 - mln::p_set, 913
 - mln::p_vertices, 927
- nslis
 - mln::geom, 258

- mln::image3d, 777
- nvalues
 - mln::doc::Value_Set, 633
 - mln::value::lut_vec, 1113
- nvertices
 - mln::p_vertices, 928
- object_id
 - mln::util::object_id, 1059
- octagon2d
 - mln::win::octagon2d, 1146
- olive
 - mln::literal, 321
- On images, 91
- On site sets, 91
- On values, 92
- one
 - mln::literal, 322
 - mln::value::int_s, 1102
 - mln::value::int_u_sat, 1107
 - mln::value::sign, 1123
- opening
 - mln::morpho::elementary, 365
- operator bool
 - mln::util::tracked_ptr, 1076
- operator decorated_image< const I, D >
 - mln::decorated_image, 593
- operator edge_id_t
 - mln::util::edge, 1037
- operator float
 - mln::value::float01, 1092
 - mln::value::float01_f, 1094
- operator graylevel< n >
 - mln::value::graylevel_f, 1099
- operator I
 - mln::plain, 931
- operator int
 - mln::value::int_s, 1102
 - mln::value::int_u_sat, 1106
 - mln::value::sign, 1123
- operator mat< n, l, U >
 - mln::algebra::h_vec, 543
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
 - mln::dpoint, 643
- operator psite
 - mln::doc::Site_Iterator, 627
- operator safe_image< const I >
 - mln::safe_image, 945
- operator sub_image< const I, S >
 - mln::sub_image, 955
- operator thru_image< const I, F >
 - mln::thru_image, 957
- operator thrubin_image< const I1, const I2, F >
 - mln::thrubin_image, 958
- operator transformed_image< const I, F >
 - mln::transformed_image, 1016
- operator typename internal::p_image_site_set< I >::ret
 - mln::p_image, 869
- operator unsigned
 - mln::value::int_u, 1104
 - mln::value::label, 1110
- operator util::tree_node< T > &
 - mln::util::branch_iter, 1029
 - mln::util::branch_iter_ind, 1031
- operator value
 - mln::doc::Value_Iterator, 631
- operator vertex_id_t
 - mln::util::vertex, 1088
- operator<
 - mln, 156, 157
 - mln::Box, 560
 - mln::Site_Set, 952
 - mln::topo, 402, 403
 - mln::util, 415
- operator<<
 - mln, 157, 158
 - mln::Box, 560
 - mln::box, 556
 - mln::fun::i2v, 242
 - mln::Gpoint, 725
 - mln::Site_Set, 952
 - mln::topo, 403, 404
 - mln::util, 415
 - mln::value, 425–428
 - mln::w_window, 1135
- operator<=
 - mln, 158, 159
 - mln::Box, 560, 561
 - mln::Site_Set, 952
- operator*
 - mln, 155
 - mln::algebra, 173
 - mln::value, 424
- operator()
 - mln::complex_image, 581
 - mln::decorated_image, 593, 594
 - mln::doc::Fastest_Image, 607, 608
 - mln::doc::Image, 616
 - mln::edge_image, 654
 - mln::extension_fun, 658
 - mln::extension_ima, 660
 - mln::extension_val, 663
 - mln::flat_image, 665
 - mln::fun::v2v::rgb8_to_rgbn, 679
 - mln::fun::x2v::bilinear, 698
 - mln::fun::x2x::linear, 701

- mln::fun::x2x::rotation, 704
- mln::fun::x2x::translation, 707
- mln::fun_image, 709
- mln::geom::complex_geometry, 721
- mln::hexa, 757
- mln::image1d, 763
- mln::image2d, 768
- mln::image2d_h, 772
- mln::image3d, 777
- mln::lazy_image, 792, 793
- mln::p2p_image, 839
- mln::p_key, 878
- mln::p_priority, 893
- mln::p_vaccess, 922
- mln::p_vertices, 928
- mln::topo::is_simple_cell, 1000
- mln::tr_image, 1014
- mln::transformed_image, 1017
- mln::unproject_image, 1018
- mln::util::array, 1025
- mln::util::greater_point, 1049
- mln::util::greater_psite, 1049
- mln::value::stack_image, 1125, 1126
- mln::value::value_array, 1127
- mln::vertex_image, 1130
- mln::violent_cast_image, 1131
- operator+
 - mln::Gpoint, 723
 - mln::topo, 401
 - mln::value, 424, 425
- operator++
 - mln, 155
 - mln::value::label, 1110
- operator+=
 - mln::Gpoint, 724
 - mln::Point, 933
 - mln::point, 939
 - mln::value::int_u_sat, 1107
- operator-
 - mln, 156
 - mln::Gpoint, 724
 - mln::topo, 402
 - mln::value, 425
 - mln::value::int_u, 1105
 - mln::Weighted_Window, 1136
- operator->
 - mln::util::tracked_ptr, 1076
- operator--
 - mln, 156
 - mln::value::label, 1110
- operator==
 - mln::Gpoint, 725
 - mln::Point, 933
 - mln::point, 939
- mln::value::int_u_sat, 1107
- operator/
 - mln::Gpoint, 725
 - mln::Point, 933
 - mln::value, 425
- operator=
 - mln::plain, 931
 - mln::util::fibonacci_heap, 1041
 - mln::util::tracked_ptr, 1077
 - mln::value::float01_f, 1094
 - mln::value::graylevel, 1097
 - mln::value::graylevel_f, 1099, 1100
 - mln::value::int_s, 1102
 - mln::value::int_u, 1105
 - mln::value::int_u_sat, 1107
 - mln::value::label, 1110, 1111
 - mln::value::proxy, 1116
 - mln::value::qt::rgb32, 1118
 - mln::value::rgb, 1120
 - mln::value::sign, 1123
- operator==
 - mln, 159, 160
 - mln::accu::stat, 171
 - mln::Box, 561
 - mln::Gpoint, 725
 - mln::Site_Set, 952
 - mln::topo, 404, 405
 - mln::util, 415, 416
 - mln::value, 428
 - mln::w_window, 1135
 - mln::window, 1154
- or_inplace
 - mln::logical, 325
- orange
 - mln::literal, 322
- ord_pair
 - mln::make, 342
- ord_strict
 - mln::util, 416
- ord_weak
 - mln::util, 416
- origin
 - mln::algebra::h_vec, 544
 - mln::literal, 322
 - mln::point, 941
- other
 - mln::util::vertex, 1088
 - mln::value, 429
- overset
 - mln::p_if, 865
- P
 - mln::graph_window_if_piter, 749
 - mln::graph_window_piter, 751

- p
 - mln::util::pix, 1063
- p2p_image
 - mln::p2p_image, 839
- p_array
 - mln::p_array, 842
- p_centered
 - mln::p_centered, 847
- p_complex
 - mln::p_complex, 850
- p_edges
 - mln::p_edges, 855, 856
- p_edges_with_mass_centers
 - mln::make, 342
- p_faces
 - mln::p_faces, 859, 860
- p_graph_piter
 - mln::p_graph_piter, 862
- p_if
 - mln::p_if, 864
- p_image
 - mln::p_image, 868
- p_indexed_bkd_piter
 - mln::p_indexed_bkd_piter, 871
- p_indexed_fwd_piter
 - mln::p_indexed_fwd_piter, 872
- p_key
 - mln::p_key, 876
- p_line2d
 - mln::p_line2d, 881
- p_mutable_array_of
 - mln::p_mutable_array_of, 885
- p_n_faces_bkd_piter
 - mln::p_n_faces_bkd_piter, 887
- p_n_faces_fwd_piter
 - mln::p_n_faces_fwd_piter, 888
- p_priority
 - mln::p_priority, 891
- p_queue
 - mln::p_queue, 897
- p_queue_fast
 - mln::p_queue_fast, 902
- p_run
 - mln::p_run, 907
- p_run2d
 - mln, 146
- p_runs2d
 - mln, 146
- p_set
 - mln::p_set, 912
- p_transformed
 - mln::p_transformed, 915
- p_transformed_piter
 - mln::p_transformed_piter, 917
- p_vaccess
 - mln::p_vaccess, 921
- p_vertices
 - mln::p_vertices, 925, 926
- p_vertices_with_mass_centers
 - mln::make, 343
- pack
 - mln::labeling, 302, 303
- pack_inplace
 - mln::labeling, 303
- pair
 - mln::util::site_pair, 1070
- parent
 - mln::util::tree_node, 1083
- paste
 - mln::data, 208
 - mln::data::impl::generic, 220
- paste_without_localization
 - mln::data, 209
- paste_without_localization_fast
 - mln::data::impl, 216
- paste_without_localization_fastest
 - mln::data::impl, 216
- paste_without_localization_lines
 - mln::data::impl, 217
- pcenter
 - mln::box, 555
- pink
 - mln::literal, 322
- piter
 - mln::box, 552
 - mln::p_array, 842
 - mln::p_centered, 846
 - mln::p_complex, 850
 - mln::p_edges, 855
 - mln::p_faces, 859
 - mln::p_if, 864
 - mln::p_image, 868
 - mln::p_key, 875
 - mln::p_line2d, 880
 - mln::p_mutable_array_of, 884
 - mln::p_priority, 891
 - mln::p_queue, 896
 - mln::p_queue_fast, 901
 - mln::p_run, 906
 - mln::p_set, 911
 - mln::p_transformed, 915
 - mln::p_vaccess, 920
 - mln::p_vertices, 925
- pix
 - mln::make, 343
 - mln::util::pix, 1063
- pixel
 - mln::make, 343

- mln::pixel, 929
- plain
 - mln::plain, 931
- plot
 - mln::draw, 234
- plus
 - mln::arith, 181, 182
 - mln::morpho, 360
- plus_cst
 - mln::arith, 182, 183
- plus_cst_inplace
 - mln::arith, 183
- plus_infty
 - mln::point, 940
- plus_inplace
 - mln::arith, 184
- pmax
 - mln::box, 555
 - mln::doc::Box, 599
- pmin
 - mln::box, 555
 - mln::doc::Box, 599
- pmin_pmax
 - mln::geom, 259
- point
 - mln::doc::Dpoint, 600
 - mln::doc::Fastest_Image, 604
 - mln::doc::Image, 614
 - mln::doc::Neighborhood, 619
 - mln::doc::Point_Site, 624
 - mln::doc::Weighted_Window, 635
 - mln::Point, 933
 - mln::point, 938, 939
- point1d
 - mln, 147
- point1df
 - mln, 147
- point2d
 - mln, 147
- point2d_h
 - mln, 147
 - mln::make, 344
- point2df
 - mln, 147
- point3d
 - mln, 147
- point3df
 - mln, 147
- point_at_index
 - mln::doc::Fastest_Image, 609
 - mln::image1d, 763
 - mln::image2d, 768
 - mln::image3d, 777
- polygon
 - mln::draw, 234
- pop
 - mln::p_priority, 893
 - mln::p_queue, 898
 - mln::p_queue_fast, 903
- pop_front
 - mln::p_priority, 894
 - mln::p_queue, 898
 - mln::p_queue_fast, 904
 - mln::util::fibonacci_heap, 1041
 - mln::util::soft_heap, 1073
- positive
 - mln::test, 394
- pred
 - mln::p_if, 865
- predicate
 - mln::p_if, 865
 - mln::test, 394
- prev
 - mln::value::label, 1111
- primary
 - mln, 161
- primary_set
 - mln::p_transformed, 916
- print
 - mln::topo::complex, 989
 - mln::util::tree_node, 1083
 - mln::window, 1153
- print_faces
 - mln::topo::complex, 989
- println
 - mln::debug, 226, 227
- println_with_border
 - mln::debug, 227
- priorities
 - mln::p_priority, 894
- propagate_if
 - mln::morpho::tree, 373
- propagate_if_value
 - mln::morpho::tree, 373
- propagate_node_to_ancestors
 - mln::morpho::tree, 374
- propagate_node_to_descendants
 - mln::morpho::tree, 374, 375
- propagate_representative
 - mln::morpho::tree, 375
- proxy
 - mln::value::proxy, 1115
- pset
 - mln::doc::Fastest_Image, 604
 - mln::doc::Image, 614
 - mln::p_vaccess, 920
- psite
 - mln::box, 552

- mln::complex_neighborhood_bkd_piter, 582
- mln::complex_neighborhood_fwd_piter, 584
- mln::complex_window_bkd_piter, 589
- mln::complex_window_fwd_piter, 590
- mln::decorated_image, 592
- mln::doc::Box, 598
- mln::doc::Fastest_Image, 605
- mln::doc::Image, 614
- mln::doc::Site_Iterator, 627
- mln::doc::Site_Set, 629
- mln::dpoint, 641
- mln::graph_elt_mixed_window, 732
- mln::graph_elt_window, 740
- mln::graph_elt_window_if, 744
- mln::hexa, 755
- mln::image2d_h, 771
- mln::interpolated, 778
- mln::p_array, 842
- mln::p_centered, 847
- mln::p_complex, 850
- mln::p_edges, 855
- mln::p_faces, 859
- mln::p_if, 864
- mln::p_image, 868
- mln::p_key, 875
- mln::p_line2d, 880
- mln::p_mutable_array_of, 884
- mln::p_priority, 891
- mln::p_queue, 897
- mln::p_queue_fast, 901
- mln::p_run, 906
- mln::p_set, 911
- mln::p_transformed, 915
- mln::p_vaccess, 920
- mln::p_vertices, 925
- mln::thruvin_image, 958
- mln::topo::is_simple_cell, 1000
- mln::tr_image, 1013
- mln::util::pix, 1062
- mln::value::stack_image, 1124
- ptransform
 - mln, 162
- purge
 - mln::p_queue_fast, 904
- purple
 - mln::literal, 322
- push
 - mln::p_priority, 894
 - mln::p_queue, 899
 - mln::p_queue_fast, 904
 - mln::util::fibonacci_heap, 1041, 1042
 - mln::util::soft_heap, 1073, 1074
- put_word
 - mln::debug, 227
- q_box
 - mln::p_line2d, 880
 - mln::p_run, 907
- qiter
 - mln::doc::Window, 638
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_elt_window_if, 744
 - mln::window, 1151
- Queue based, 110
- r_element
 - mln::p_image, 868
 - mln::p_key, 876
 - mln::p_set, 911
- rag_and_labeled_wsl
 - mln::make, 344
- rank_filter
 - mln::morpho, 361
- read_header
 - mln::io::fld, 274
- rectangle2d
 - mln::win::rectangle2d, 1147
- rectangularity
 - mln::accu::site_set::rectangularity, 506
- red
 - mln::literal, 322
 - mln::value::qt::rgb32, 1118
 - mln::value::rgb, 1120
- region_adjacency_graph
 - mln::make, 345
- regional_maxima
 - mln::labeling, 303
- regional_minima
 - mln::labeling, 304
- registration1
 - mln::registration, 387
- registration2
 - mln::registration, 388
- registration3
 - mln::registration, 388
- regular
 - mln::window, 1151
- relabel
 - mln::labeled_image, 786
 - mln::labeled_image_base, 789, 790
 - mln::labeling, 304, 305
- relabel_inplace
 - mln::labeling, 305
- relabelfun
 - mln::make, 345
- remove
 - mln::p_image, 869
 - mln::p_key, 878

- mln::p_set, [913](#)
 - mln::util::set, [1069](#)
- remove_key
 - mln::p_key, [878](#)
- replace
 - mln::data, [209](#)
- reserve
 - mln::p_array, [844](#)
 - mln::p_mutable_array_of, [886](#)
 - mln::p_queue_fast, [904](#)
 - mln::topo::n_faces_set, [1008](#)
 - mln::util::array, [1026](#)
- resize
 - mln::border, [191](#)
 - mln::p_array, [844](#)
 - mln::util::array, [1026](#)
- result
 - mln::graph::attribute::card_t, [727](#)
 - mln::graph::attribute::representative_t, [728](#)
 - mln::topo::is_simple_cell, [1000](#)
 - mln::util::array, [1022](#)
- revert
 - mln::arith, [184](#)
- revert_inplace
 - mln::arith, [184](#)
- rgb
 - mln::value::rgb, [1119](#)
- rgb16
 - mln::value, [424](#)
- rgb32
 - mln::value::qt::rgb32, [1117](#)
- rgb8
 - mln::value, [424](#)
- rgb8_2complex_image3df
 - mln, [147](#)
- root
 - mln::util::tree, [1079](#)
- rotate
 - mln::geom, [259](#), [260](#)
- rotation
 - mln::fun::x2x::rotation, [703](#), [704](#)
- Routines, [99](#)
- run_length
 - mln::box_runend_piter, [562](#)
 - mln::box_runstart_piter, [564](#)
- rvalue
 - mln::complex_image, [580](#)
 - mln::decorated_image, [593](#)
 - mln::doc::Fastest_Image, [605](#)
 - mln::doc::Generalized_Pixel, [610](#)
 - mln::doc::Image, [614](#)
 - mln::doc::Pixel_Iterator, [622](#)
 - mln::extension_fun, [657](#)
 - mln::extension_ima, [659](#)
 - mln::extension_val, [662](#)
 - mln::flat_image, [664](#)
 - mln::fun_image, [708](#)
 - mln::hexa, [756](#)
 - mln::image1d, [761](#)
 - mln::image2d, [766](#)
 - mln::image2d_h, [771](#)
 - mln::image3d, [774](#)
 - mln::interpolated, [779](#)
 - mln::lazy_image, [792](#)
 - mln::thruvin_image, [958](#)
 - mln::tr_image, [1013](#)
 - mln::value::stack_image, [1125](#)
 - mln::violent_cast_image, [1131](#)
- S
 - mln::p_image, [868](#)
- sagittal_dec
 - mln, [162](#)
- saturate
 - mln::data, [209](#)
- saturate_inplace
 - mln::data, [210](#)
- save
 - mln::io::cloud, [269](#)
 - mln::io::dump, [272](#)
 - mln::io::magick, [275](#)
 - mln::io::off, [277](#)
 - mln::io::pbm, [278](#)
 - mln::io::pfm, [281](#)
 - mln::io::pgm, [282](#)
 - mln::io::plot, [284](#), [285](#)
 - mln::io::pnm, [286](#)
 - mln::io::ppm, [289](#)
 - mln::io::raw, [291](#)
 - mln::io::txt, [292](#)
- save_bin_alt
 - mln::io::off, [277](#)
- search
 - mln::util::tree_node, [1083](#)
- search_rec
 - mln::util::tree_node, [1083](#)
- second
 - mln::accu::pair, [497](#)
 - mln::accu::stat::min_max, [525](#)
 - mln::util::couple, [1033](#)
 - mln::util::ord_pair, [1061](#)
 - mln::util::site_pair, [1071](#)
- second_accu
 - mln::accu::pair, [497](#)
 - mln::accu::stat::min_max, [525](#)
- seeds2tiling
 - mln::geom, [260](#)
 - mln::geom::impl, [263](#)

- seeds2tiling_roundness
 - mln::geom, 261
- segment1d
 - modwin1d, 112
- set
 - mln::util::set, 1066
- set_all
 - mln::dpoint, 643
 - mln::point, 940
- set_alpha
 - mln::fun::x2x::rotation, 704
- set_axis
 - mln::fun::x2x::rotation, 704
- set_cplx
 - mln::topo::algebraic_face, 977
 - mln::topo::algebraic_n_face, 982
 - mln::topo::face, 993
 - mln::topo::n_face, 1004
- set_face_id
 - mln::topo::algebraic_face, 977
 - mln::topo::algebraic_n_face, 982
 - mln::topo::face, 993
 - mln::topo::n_face, 1004
- set_image
 - mln::topo::is_simple_cell, 1001
- set_n
 - mln::topo::algebraic_face, 977
 - mln::topo::face, 993
- set_nbits
 - mln::value::float01, 1093
- set_parent
 - mln::util::tree_node, 1084
- set_sign
 - mln::topo::algebraic_face, 977
 - mln::topo::algebraic_n_face, 982
- set_t
 - mln::fun::x2x::translation, 707
- set_tr
 - mln::tr_image, 1015
- set_value
 - mln::accu::count_adjacent_vertices, 437
 - mln::accu::count_labels, 439
 - mln::accu::count_value, 440
 - mln::accu::math::count, 452
 - mln::accu::shape::height, 502
 - mln::accu::shape::volume, 505
 - mln::accu::stat::max, 512
 - mln::accu::stat::min, 521
 - mln::morpho::attribute::sum, 832
- sign
 - mln::topo::algebraic_face, 978
 - mln::topo::algebraic_n_face, 982
 - mln::value::sign, 1122
- site
 - mln::box, 552
 - mln::doc::Box, 598
 - mln::doc::Site_Set, 629
 - mln::dpoint, 641
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_elt_window_if, 744
 - mln::graph_window_base, 747
 - mln::p_centered, 847
 - mln::tr_image, 1013
- Site sets, 108
- site_function_t
 - mln::edge_image, 653
 - mln::vertex_image, 1129
- site_set
 - mln::complex_psite, 587
 - mln::draw, 235
- size
 - mln::util::array, 1026
 - mln::win::line, 1144
 - mln::window, 1153
- skeleton
 - mln::complex_image, 580
 - mln::decorated_image, 593
 - mln::doc::Fastest_Image, 605
 - mln::doc::Image, 614
 - mln::edge_image, 654
 - mln::extended, 655
 - mln::extension_fun, 657
 - mln::extension_ima, 659
 - mln::extension_val, 662
 - mln::flat_image, 664
 - mln::fun_image, 708
 - mln::hexa, 756
 - mln::image1d, 761
 - mln::image2d, 766
 - mln::image2d_h, 771
 - mln::image3d, 774
 - mln::interpolated, 779
 - mln::labeled_image, 785
 - mln::lazy_image, 792
 - mln::p2p_image, 838
 - mln::plain, 931
 - mln::pw::image, 943
 - mln::safe_image, 945
 - mln::sub_image, 954
 - mln::sub_image_if, 956
 - mln::thruvin_image, 958
 - mln::tr_image, 1014
 - mln::transformed_image, 1016
 - mln::value::stack_image, 1125
 - mln::vertex_image, 1129
 - mln::violent_cast_image, 1131
- slices_2d

- mln::debug, 227
- sline3d
 - modwin3d, 116
- soft_heap
 - mln::util::soft_heap, 1072
- sort_offsets_increasing
 - mln::data, 210
- sort_psites_decreasing
 - mln::data, 210
- sort_psites_increasing
 - mln::data, 210
- space_2complex_geometry
 - mln, 148
- Sparse types, 109
- sphere3d
 - modwin3d, 116
- sqr_l2
 - mln::norm, 382
- stack
 - mln::value, 429
- stack_image
 - mln::value::stack_image, 1125
- standard_deviation
 - mln::accu::stat::variance, 534
- start
 - mln::doc::Iterator, 618
 - mln::doc::Pixel_Iterator, 623
 - mln::doc::Site_Iterator, 627
 - mln::doc::Value_Iterator, 631
 - mln::dpoints_bkd_pixter, 646
 - mln::dpoints_fwd_pixter, 648
 - mln::p_run, 909
 - mln::topo::face_bkd_iter, 994
 - mln::topo::face_fwd_iter, 996
 - mln::topo::n_face_bkd_iter, 1006
 - mln::topo::n_face_fwd_iter, 1007
 - mln::topo::static_n_face_bkd_iter, 1010
 - mln::topo::static_n_face_fwd_iter, 1012
 - mln::util::branch_iter, 1029
 - mln::util::branch_iter_ind, 1031
- static_n_face_bkd_iter
 - mln::topo::static_n_face_bkd_iter, 1010
- static_n_face_fwd_iter
 - mln::topo::static_n_face_fwd_iter, 1011
- std_deque
 - mln::p_queue, 899
- std_vector
 - mln::p_array, 845
 - mln::p_line2d, 882
 - mln::p_queue_fast, 904
 - mln::p_set, 913
 - mln::util::array, 1026
 - mln::util::set, 1069
 - mln::w_window, 1134
 - mln::win::rectangle2d, 1148
 - mln::window, 1153
- stretch
 - mln::data, 211
 - mln::data::impl, 217
- structural
 - mln::morpho::closing::approx, 364
 - mln::morpho::opening::approx, 367
- sub_image
 - mln::sub_image, 954
- sub_image_if
 - mln::sub_image_if, 956
- subdomain
 - mln::labeled_image, 787
 - mln::labeled_image_base, 790
- subsampling
 - mln::subsampling, 392
- subtractive
 - mln::morpho::tree::filter, 377
- sum
 - mln::accu::stat::mean, 515
 - mln::accu::stat::variance, 535
 - mln::estim, 237
- superpose
 - mln::debug, 227, 228
 - mln::labeling, 306
 - mln::morpho::watershed, 379
- sym
 - mln::doc::Weighted_Window, 636
 - mln::graph_elt_mixed_window, 733
 - mln::graph_elt_window, 741
 - mln::graph_elt_window_if, 746
 - mln::graph_window_base, 748
 - mln::w_window, 1134
 - mln::win, 431
 - mln::window, 1154
- sym_diff
 - mln::Box, 561
 - mln::Site_Set, 953
- t
 - mln::algebra::h_mat, 542
 - mln::algebra::h_vec, 543
 - mln::fun::x2x::translation, 707
- take
 - mln::accu, 165
 - mln::accu::histo, 442
 - mln::accu::label_used, 443
 - mln::accu::stat::histo3d_rgb, 510
 - mln::accu::stat::median_alt, 517
 - mln::doc::Accumulator, 596
- take_as_init
 - mln::accu::center, 434
 - mln::accu::convolve, 436

- mln::accu::count_adjacent_vertices, 437
- mln::accu::count_labels, 439
- mln::accu::count_value, 440
- mln::accu::histo, 442
- mln::accu::label_used, 443
- mln::accu::logic::land, 445
- mln::accu::logic::land_basic, 446
- mln::accu::logic::lor, 448
- mln::accu::logic::lor_basic, 449
- mln::accu::maj_h, 451
- mln::accu::math::count, 452
- mln::accu::math::inf, 454
- mln::accu::math::sum, 455
- mln::accu::math::sup, 457
- mln::accu::max_site, 458
- mln::accu::nil, 493
- mln::accu::p, 495
- mln::accu::pair, 498
- mln::accu::rms, 499
- mln::accu::shape::bbox, 500
- mln::accu::shape::height, 503
- mln::accu::shape::volume, 505
- mln::accu::site_set::rectangularity, 506
- mln::accu::stat::deviation, 508
- mln::accu::stat::histo3d_rgb, 511
- mln::accu::stat::max, 512
- mln::accu::stat::max_h, 514
- mln::accu::stat::mean, 515
- mln::accu::stat::median_alt, 517
- mln::accu::stat::median_h, 519
- mln::accu::stat::min, 521
- mln::accu::stat::min_h, 523
- mln::accu::stat::min_max, 525
- mln::accu::stat::rank, 527
- mln::accu::stat::rank < bool >, 529
- mln::accu::stat::rank_high_quant, 530
- mln::accu::stat::var, 532
- mln::accu::stat::variance, 535
- mln::accu::tuple, 536
- mln::accu::val, 538
- mln::Accumulator, 540
- mln::morpho::attribute::card, 825
- mln::morpho::attribute::count_adjacent_vertices, 827
- mln::morpho::attribute::height, 828
- mln::morpho::attribute::sharpness, 830
- mln::morpho::attribute::sum, 832
- mln::morpho::attribute::volume, 834
- take_n_times
 - mln::accu::center, 434
 - mln::accu::convolve, 436
 - mln::accu::count_adjacent_vertices, 437
 - mln::accu::count_labels, 439
 - mln::accu::count_value, 441
 - mln::accu::histo, 442
 - mln::accu::label_used, 444
 - mln::accu::logic::land, 445
 - mln::accu::logic::land_basic, 447
 - mln::accu::logic::lor, 448
 - mln::accu::logic::lor_basic, 449
 - mln::accu::maj_h, 451
 - mln::accu::math::count, 452
 - mln::accu::math::inf, 454
 - mln::accu::math::sum, 455
 - mln::accu::math::sup, 457
 - mln::accu::max_site, 458
 - mln::accu::nil, 493
 - mln::accu::p, 495
 - mln::accu::pair, 498
 - mln::accu::rms, 499
 - mln::accu::shape::bbox, 501
 - mln::accu::shape::height, 503
 - mln::accu::shape::volume, 505
 - mln::accu::site_set::rectangularity, 507
 - mln::accu::stat::deviation, 508
 - mln::accu::stat::histo3d_rgb, 511
 - mln::accu::stat::max, 512
 - mln::accu::stat::max_h, 514
 - mln::accu::stat::mean, 516
 - mln::accu::stat::median_alt, 517
 - mln::accu::stat::median_h, 519
 - mln::accu::stat::min, 521
 - mln::accu::stat::min_h, 523
 - mln::accu::stat::min_max, 526
 - mln::accu::stat::rank, 527
 - mln::accu::stat::rank < bool >, 529
 - mln::accu::stat::rank_high_quant, 530
 - mln::accu::stat::var, 532
 - mln::accu::stat::variance, 535
 - mln::accu::tuple, 537
 - mln::accu::val, 538
 - mln::Accumulator, 540
 - mln::morpho::attribute::card, 825
 - mln::morpho::attribute::count_adjacent_vertices, 827
 - mln::morpho::attribute::height, 829
 - mln::morpho::attribute::sharpness, 831
 - mln::morpho::attribute::sum, 832
 - mln::morpho::attribute::volume, 834
- target
 - mln::graph_elt_mixed_window, 732
 - mln::graph_elt_window, 740
 - mln::graph_elt_window_if, 744
- target_site_set
 - mln::graph_window_piter, 753
- teal
 - mln::literal, 322
- the

- mln::value::set, 1121
- thick_miss
 - mln::morpho, 361
- thickening
 - mln::morpho, 361
- thin_fit
 - mln::morpho, 361
- thinning
 - mln::morpho, 361
- threshold
 - mln::binarization, 187
- times
 - mln::arith, 185
- times_cst
 - mln::arith, 185
- times_inplace
 - mln::arith, 185
- to
 - mln::convert, 199
- to_dpoint
 - mln::convert, 199
 - mln::Dpoint, 639
- to_enc
 - mln::data, 211
- to_float
 - mln::value::graylevel, 1097
- to_fun
 - mln::convert, 199, 202
- to_h_vec
 - mln::point, 940
- to_image
 - mln::convert, 199
- to_larger
 - mln::box, 555
- to_nbits
 - mln::value::float01, 1093
- to_neighb
 - mln::graph, 264
- to_p_array
 - mln::convert, 200
- to_p_set
 - mln::convert, 200, 201
- to_point
 - mln::doc::Point_Site, 625
 - mln::Point, 933
- to_qimage
 - mln::convert, 201
- to_result
 - mln::accu::center, 434
 - mln::accu::convolve, 436
 - mln::accu::count_adjacent_vertices, 438
 - mln::accu::count_labels, 439
 - mln::accu::count_value, 441
 - mln::accu::label_used, 444
 - mln::accu::logic::land, 445
 - mln::accu::logic::land_basic, 447
 - mln::accu::logic::lor, 448
 - mln::accu::logic::lor_basic, 450
 - mln::accu::maj_h, 451
 - mln::accu::math::count, 453
 - mln::accu::math::inf, 454
 - mln::accu::math::sum, 455
 - mln::accu::math::sup, 457
 - mln::accu::max_site, 458
 - mln::accu::nil, 493
 - mln::accu::p, 495
 - mln::accu::pair, 498
 - mln::accu::rms, 499
 - mln::accu::shape::bbox, 501
 - mln::accu::shape::height, 503
 - mln::accu::shape::volume, 505
 - mln::accu::site_set::rectangularity, 507
 - mln::accu::stat::deviation, 508
 - mln::accu::stat::histo3d_rgb, 511
 - mln::accu::stat::max, 512
 - mln::accu::stat::max_h, 514
 - mln::accu::stat::mean, 516
 - mln::accu::stat::median_alt, 517
 - mln::accu::stat::median_h, 519
 - mln::accu::stat::min, 522
 - mln::accu::stat::min_h, 523
 - mln::accu::stat::min_max, 526
 - mln::accu::stat::rank, 527
 - mln::accu::stat::rank < bool >, 529
 - mln::accu::stat::rank_high_quant, 530
 - mln::accu::stat::var, 532
 - mln::accu::stat::variance, 535
 - mln::accu::tuple, 537
 - mln::accu::val, 538
 - mln::morpho::attribute::card, 826
 - mln::morpho::attribute::count_adjacent_vertices, 827
 - mln::morpho::attribute::height, 829
 - mln::morpho::attribute::sharpness, 831
 - mln::morpho::attribute::sum, 833
 - mln::morpho::attribute::volume, 834
- to_upper_window
 - mln::convert, 201
- to_value
 - mln::value::proxy, 1116
- to_vec
 - mln::algebra::h_vec, 544
 - mln::dpoint, 643
 - mln::point, 940
- to_win
 - mln::graph, 265
- to_window
 - mln::convert, 201, 202

- toggle
 - mln::p_image, 870
- top_hat_black
 - mln::morpho, 362
 - mln::morpho::elementary, 365
- top_hat_self_complementary
 - mln::morpho, 362
 - mln::morpho::elementary, 365
- top_hat_white
 - mln::morpho, 362
 - mln::morpho::elementary, 366
- topological
 - mln::morpho::watershed, 380
- tr
 - mln::tr_image, 1015
- tr_image
 - mln::tr_image, 1014
- tracked_ptr
 - mln::util::tracked_ptr, 1076
- trait::graph, 1155
- trait::graph< mln::complex_image< 1, G, V > >, 1156
- trait::graph< mln::image2d< T > >, 1156
- transform
 - mln::data, 211, 212
 - mln::data::impl::generic, 220
- transform_inplace
 - mln::data, 212
 - mln::data::impl::generic, 221
- transform_inplace_lowq
 - mln::data::impl, 218
- transformed_image
 - mln::transformed_image, 1016
- translate
 - mln::geom, 261, 262
- translation
 - mln::fun::x2x::translation, 706
- tree
 - mln::util::tree, 1078
- tree_fast_to_image
 - mln::util, 416
- tree_node
 - mln::util::tree_node, 1081
- tree_to_fast
 - mln::util, 416
- tree_to_image
 - mln::util, 417
- Types, 98
- uni
 - mln::Box, 561
 - mln::Site_Set, 953
- unique
 - mln::Box, 561
 - mln::Site_Set, 953
- unproject_image
 - mln::unproject_image, 1018
- unsigned_2complex_image3df
 - mln, 148
- untake
 - mln::morpho::attribute::sum, 833
- up
 - mln, 162
- update
 - mln::data, 213
 - mln::data::impl::generic, 221
 - mln::dpoints_bkd_pixter, 646
 - mln::dpoints_fwd_pixter, 649
- update_data
 - mln::labeled_image, 787
 - mln::labeled_image_base, 790
- update_fastest
 - mln::data::impl, 218
- update_id
 - mln::util::edge, 1037
 - mln::util::vertex, 1088
- util_set
 - mln::p_set, 913
- util_tree
 - mln::util::branch, 1028
- Utilities, 110
- v
 - mln::util::pix, 1063
- v1
 - mln::util::edge, 1037
 - mln::util::graph, 1047
 - mln::util::line_graph, 1055
- v2
 - mln::util::edge, 1038
 - mln::util::graph, 1047
 - mln::util::line_graph, 1055
- v2w2v functions, 118
- v2w_w2v functions, 119
- v_ith_nbh_edge
 - mln::util::graph, 1047
 - mln::util::line_graph, 1055
- v_ith_nbh_vertex
 - mln::util::graph, 1048
 - mln::util::line_graph, 1056
- v_nmax
 - mln::util::graph, 1048
 - mln::util::line_graph, 1056
- v_other
 - mln::util::edge, 1038
- val
 - mln::doc::Generalized_Pixel, 611
 - mln::doc::Pixel_Iterator, 623

- value
 - mln::accu::shape::height, 502
 - mln::accu::shape::volume, 504
 - mln::complex_image, 580
 - mln::doc::Fastest_Image, 605
 - mln::doc::Generalized_Pixel, 610
 - mln::doc::Image, 615
 - mln::doc::Pixel_Iterator, 622
 - mln::doc::Value_Iterator, 631
 - mln::doc::Value_Set, 633
 - mln::extended, 655
 - mln::extension_fun, 657
 - mln::extension_ima, 660
 - mln::extension_val, 662
 - mln::flat_image, 664
 - mln::fun_image, 709
 - mln::hexa, 756
 - mln::image1d, 761
 - mln::image2d, 766
 - mln::image2d_h, 771
 - mln::image3d, 774
 - mln::interpolated, 779
 - mln::labeling, 306
 - mln::p_vaccess, 920
 - mln::thruin_image, 958
 - mln::tr_image, 1014
 - mln::util::pix, 1062
 - mln::value::float01, 1093
 - mln::value::float01_f, 1094
 - mln::value::graylevel, 1097
 - mln::value::graylevel_f, 1100
 - mln::value::lut_vec, 1112
 - mln::value::stack_image, 1125
 - mln::violent_cast_image, 1131
- value_and_compute
 - mln::labeling, 306
- value_array
 - mln::value::value_array, 1127
- value_ind
 - mln::value::float01, 1093
- value_t
 - mln::util::object_id, 1059
- values
 - mln::complex_image, 581
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 616
 - mln::p_vaccess, 922
- Values morphers, 96
- var
 - mln::accu::stat::variance, 535
- variance
 - mln::accu::stat::var, 533
- vbbox
 - mln::image1d, 764
 - mln::image3d, 777
- vec
 - mln::dpoint, 641
 - mln::make, 346, 347
 - mln::point, 938
- vec2d_d
 - mln, 148
- vec2d_f
 - mln, 148
- vec3d_d
 - mln, 148
- vec3d_f
 - mln, 148
- vect
 - mln::accu::histo, 442
- vertex
 - mln::p_vertices, 925
 - mln::util::graph, 1048
 - mln::util::line_graph, 1056
 - mln::util::vertex, 1086
- vertex_id_t
 - mln::util, 414
- vertex_image
 - mln::make, 347
 - mln::vertex_image, 1129
- vertex_nbh_edge_fwd_iter
 - mln::util::graph, 1044
 - mln::util::line_graph, 1053
- vertex_nbh_vertex_fwd_iter
 - mln::util::graph, 1044
 - mln::util::line_graph, 1053
- vertical_symmetry
 - mln::geom, 262
- vertices_t
 - mln::util::graph, 1045
 - mln::util::line_graph, 1053
- violent_cast_image
 - mln::violent_cast_image, 1131
- violet
 - mln::literal, 322
- vline2d
 - modwin2d, 114
- volume
 - mln::morpho::attribute::sharpness, 831
 - mln::win::cuboid3d, 1141
- voronoi
 - mln::make, 348
- vprod
 - mln::algebra, 173
- vset
 - mln::doc::Fastest_Image, 605
 - mln::doc::Image, 615
 - mln::p_vaccess, 920
 - mln::value::value_array, 1127

vv2b functions, 119

w

- mln::w_window, 1134

w_window

- mln::make, 348
- mln::w_window, 1134

w_window1d

- mln::make, 348

w_window2d

- mln::make, 349

w_window3d

- mln::make, 349

w_window_directional

- mln::make, 349

weight

- mln::doc::Weighted_Window, 636
- mln::w_window, 1133

weights

- mln::w_window, 1135

white

- mln::literal, 323

width

- mln::win::cuboid3d, 1142
- mln::win::rectangle2d, 1148

win

- mln::doc::Weighted_Window, 636
- mln::w_window, 1135

win_c4p

- modwin2d, 115

win_c4p_3d

- modwin3d, 117

win_c8p

- modwin2d, 115

win_c8p_3d

- modwin3d, 117

win_t

- mln::edge_image, 654
- mln::vertex_image, 1129

window

- mln::doc::Weighted_Window, 636
- mln::p_centered, 848
- mln::window, 1151

window1d

- modwin1d, 113

window2d

- modwin2d, 114

window3d

- modwin3d, 117

Windows, 111

wrap

- mln::data, 213
- mln::labeling, 307

write_header

- mln::io::fld, 274

xor_inplace

- mln::logical, 326

yellow

- mln::literal, 323

z_order

- mln::debug, 228

zero

- mln::algebra::h_vec, 544
- mln::literal, 323
- mln::value::int_s, 1102
- mln::value::int_u_sat, 1107
- mln::value::qt::rgb32, 1118
- mln::value::rgb, 1120
- mln::value::sign, 1123